D2.1: REWIRE Operational Landscape, Requirements, and Reference Architecture - Initial Version

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/ D2.1 / v1.0
Work package:	WP 2
Due Date:	30/09/2023
Actual Submission Date:	08/11/2023
Responsible Organisation:	UBITECH
Editor:	Dr. Dimitris Papamartzivanos,
Dissemination Level:	PU
Revision:	v1.0
Abstract:	This deliverable presents the reference architecture of the REWIRE framework. It performs a state of the art analysis of the main technological pillars of the project, it defines the methodology of the REWIRE MVP and then the reference architecture is introduced. The deliverable analyses the two operational phases of the architecture, i.e.., the design-time and runtime phases, while it provides details on each component of the architecture individually. In addition, the use cases and the user stories to be validated are documented, while a detailed analysis of the framework requirements is provided.
Keywords:	Reference architecture, use cases, user stories, KPIs, MVP, requirements,

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

Versioning and contribution history

Version	Date	Author	Notes
0.1	20/02/2023	Dr. Dimitris Papamartzivanos (UBITECH)	ToC
0.2	20/03/23	UBITECH, UCL, Collins, NEC, SECURA, TUD, SURREY	Chapter 2 – State of the art analysis
0.25	31/03/2023	All partners	Chapter 5 – Initial definition of requirements
0.3	21/04/2023	Dr. Sofianna Menesidou (UBITECH)	Chapter 3 - Methodology
0.4	30/06/2022	All partners	Chapter 4 – Conceptual architecture and description on individual components Chapter 5 – Refinements on Requirements
0.45	14/07/2023	Use case partners (LSF, KEN, ODINS), UBITECH	Chapter 6 – Definition of use cases and user stories
0.5	04/08/2023	SURREY, UCL, Collins teams	Chapter 7 and 8 – Definition of threat landscape and lifecycle management
0.6	22/08/2023	Dimitris Papamartzivanos, Thanasis Giannetsos, Sofianna Menesidou (UBITECH)	Revision and comments for improvement
0.7	22/09/2023	All partners	Updates to all chapters
0.75	06/10/2023	Dr. Thanasis Giannetsos (UBITECH)	Revision and comments for improvement
0.8	20/10/2023	All partners	Updates to all chapters
0.9	25/10/2023	Dr. Sofianna Menesidou (UBITECH)	Refinements to all chapters, Release to Consortium for internal review
0.95	01/11/2023	UCL, Collins, KEN, ODINS, NEC	Internal review process
0.98	07/11/2023	Dr. Dimitris Papamartzivanos (UBITECH)	Final version.
1.0	08/11/2023	Thanasis Charemis (UBITECH)	Final submission

Executive Summary

D2.1 provides the **conceptual REWIRE architecture** and the technical **requirements** along with the requirements of the three use cases that will be used for validation purposes. In this context, it provides a detailed view of REWIRE's use cases and scenarios and describes **13 user stories** to be used for validation of core REWIRE pillars (e.g., the REWIRE customisable TEE). In addition, an initial set of KPIs per use case has been devised (to be revised under WP6, T6.1), hinting at business and technical indicators that will be tested under the piloting WP of the project.

The overall purpose of D2.1 is to provide a reference document for the REWIRE project to be used as input for the technical WPs. In parallel, the consortium has worked towards defining the technical requirements of the REWIRE needed for fulfilling the project's vision of providing a holistic security management framework that can safeguard IoT environments during the whole spectrum of their lifecycle, i.e., from the Design to the Runtime phases, capitalizing on trust-aware defense mechanisms that exploit emerging technologies based on Formal verification, Theorem Proofs, Open Standard Instruction Set Architectures (ISA), Trusted Computing, Blockchain and Artificial Intelligence (AI). Requirements, with a holistic view on a security solution in today's IoT ecosystems, have been categorised as mandatory and desirable, and in total, 31 technical requirements have been extracted.

Following these activities, a mapping between the project's technical requirements and the use case requirements has been conducted, identifying the exact needs of each use case and the specific REWIRE functionality that will be tested in each scenario. That mapping has also allowed the consortium to devise the REWIRE MVP. This will cover the needs of the use cases as well as other horizontal requirements that are necessary for demonstrating the overall REWIRE concept.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Scope and Purpose

The vision of REWIRE is to provide a holistic security management framework for safeguarding the next-generation connected "Systems-of-Systems" (SoS). More specifically, the rationale is to capitalize on trust-aware defence mechanisms that exploit emerging technologies based on Formal verification, Theorem Proofs, Open Standard Instruction Set Architectures (ISA), Trusted Computing, Blockchain, and Artificial Intelligence (AI). Thus, the goal of the REWIRE is to offer a harmonized toolchain to efficiently protect IoT deployments during their entire lifecycle. The deliverable at hand provides an introduction to the REWIRE's scope and vision, denoting the technical requirements that have been identified by the partners, and the use cases for the implementation of the envisaged REWIRE scalable and multifunctional cybersecurity platform. The deliverable provides in detail the REWIRE conceptual architecture, the technical requirements of the project, some representative reference scenarios, and a number of user stories to be investigated within these scenarios.

1.2 Relation to other WPs and Deliverables

As the reference architecture and technical requirements deliverable, this deliverable serves as the basis for all later WPs and deliverables. This first initial version of the deliverable includes the complete documentation of the REWIRE requirements of T2.1, the REWIRE reference architecture of T2.2, and the progress of the rest of the tasks in the WP (e.g., T2.3, T2.4, and T2.5) until M12. The final version of this deliverable will be provided during M21 and will include the complete developments of T2.3-T2.5 and potential updates on the reference architecture. In addition, D2.1 constitutes the baseline for Milestone MS1 - Availability of the conceptual architecture, operational landscape, and requirements to be met by the REWIRE framework, as it delivers the architectural specification of the REWIRE architecture. The inter-dependencies among the REWIRE WPs are shown in the Pert diagram, illustrated in Figure 1 below.

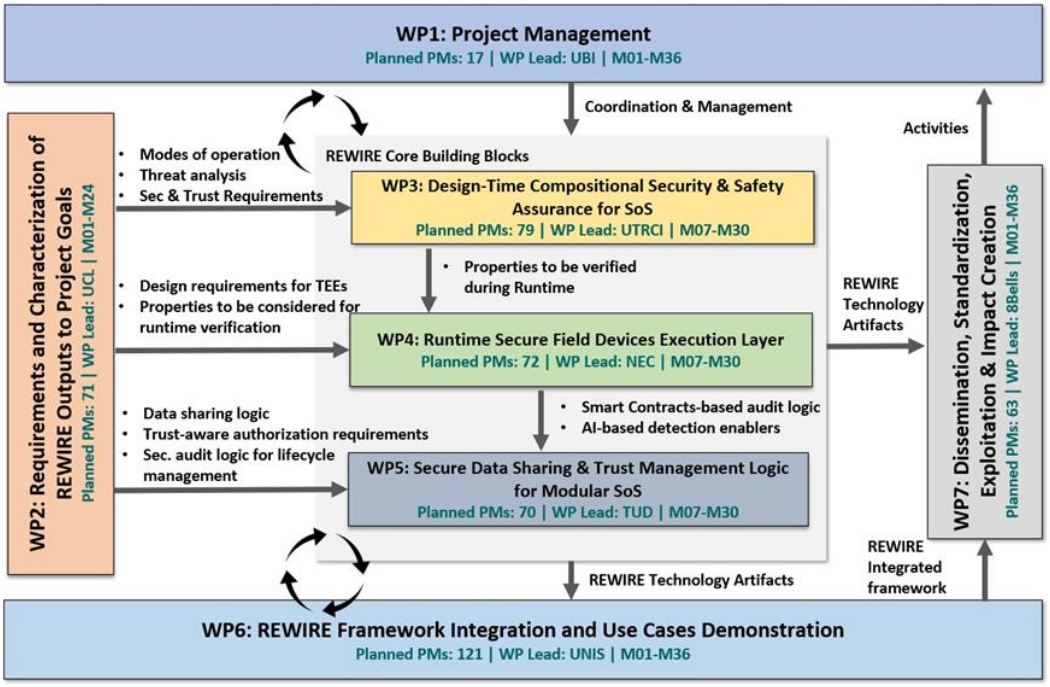

Figure 1.1: REWIRE Pert diagram

1.3 Deliverable Structure

In **Chapter 2** we describe the background and state-of-the-art analysis regarding flexible hardware architectures, formal verification, trust governance, runtime monitoring, identity management, SW/FW verification, certification through blockchain, and AI-based threat detection. The overall methodology of pf defining the REWIRE MVP is presented in **Chapter 3. Chapter 4** describes in detail the conceptual architecture including the functional components and the core functional objectives of the project. **Chapter 5** serves with detailed lists of the technical requirements (e.g., functional, security, operational assurance, formal verification, and RoT requirements). **Chapter 6** outlines the three use cases in detail, including the reference scenarios adopted by each demonstrator. The user stories from which the requirements will be drawn as part of the scenarios are described in detail and will govern the industrial demonstrations of the REWIRE. **Chapter 7** describes the trust indicators and the threat landscape analysis in order to set the scene for the underlying threats to be considered in REWIRE. **Chapter 8** documents aspects relevant to managing the security lifecycle and monitoring the secure on-boarding of the devices. Finally, **Chapter 9** maps the user stories to the technical requirements and defines the scope of the project's demonstrator (MVP), and **Chapter 10** concludes the deliverable.

Chapter 2
REWIRE Vision and Background

This chapter provides the mission of REWIRE project, the core underlying technological pillars and the consortium's vision on to enhancing the security posture of next-generation SoS. Thus, the chapter set the scene for the REWIRE conceptual architecture and the functional requirements.

2.1 REWIRE Mission and Underlying Technologies

REWIRE's mission is to define a sound methodological (aligned with existing state-of-the-art methods), technological (aligned with current cybersecurity enablers, IoT-related protection techniques), and organizational (aligned with current skill development stakeholders' involvement) approach towards securing IoT devices under a zero-trust concept throughout the entire lifecycle. The REWIRE approach focuses on developing a novel security and trust assessment framework, for Next-Generation connected "Systems-of-Systems" (SoS) covering the strict security, safety, and resilience requirements during the entire lifecycle of a CPS. This approach enables real-time protection and implements a continuous security improvement process, from the Design Phase, covering the extraction of the level of trustworthiness per device based on overarching system requirements, to the Runtime Phase, covering the operation, update re-configuration, and even decommissioning of a compromised device.

REWIRE addresses some of the most challenging open issues of cybersecurity research in the ICT domain that based on its use cases will transform (among others) smart cities, automotive, and satellite communications; those domains are tightly related to the main emerging trends of the IoT and embedded security field. More specifically, REWIRE will capitalise on trust-aware defence mechanisms that exploit emerging technologies based on Formal verification, Theorem Proofs, Open Standard Instruction Set Architectures (ISA), Trusted Computing, Blockchain, and Artificial Intelligence (AI). Thus, REWIRE will have to address critical threats to realise the business and economic potential of these emerging technologies in the context of the aforementioned domains. A challenging issue that the REWIRE project will tries to solve is how to achieve the Zero-Trust and security-by-design principles under the concept of "Never Trust, Always Verify". By coupling the Zero-Trust and security-by-design principles under the concept of "Never Trust, Always Verify", REWIRE will ensure security throughout continuous security auditing, trust computing, and theorem proofs for defining open ISA microarchitectures for reducing security threats of open-source hardware and software for connected devices.

Enabling those promising capabilities above and addressing the associated challenges of doing so, has not been sufficiently researched and validated and is one of the primary goals of the REWIRE project for enhancing the security posture of next-generation SoS. The core value propositions of REWIRE are in the fields of theorem proving and formal verification, attestation, dynamic real-time risk assessment, misbehaviour and vulnerability detection, and enhanced certification for the device state through the use of Blockchain structures. In this context, REWIRE will investigate, in the following sections, the adoption of these key technologies, in the above-mentioned fields.

2.2 REWIRE Research Pillars and State-of-the-art analysis

This section is focused on the main research pillars of REWIRE project, providing the state-of-the-art analysis along with methods and tools used in the literature as well as the corresponding issues and open challenges that needs to be addresses in the context of REWIRE.

2.2.1. Design of flexible hardware architectures converging Security and Performance

One core research question that REWIRE tries to answer is how to design flexible hardware architectures in an efficient way to reduce security threats, as the first defence mechanism from the very beginning of a system's life cycle. The first step towards this direction is to model a wide range of security requirements of the hardware itself and then formally verify these security requirements. Thus, system designers and/or developers will be in a position to detect potential vulnerabilities in the system and analyse how they would propagate in the architecture before building it. A formalized methodology that is used to support the requirements, design, analysis, verification, and validation associated with the development of complex systems is Model Based Engineering (MBE). MBE is at the core of the toolchain envisioned in REWIRE. Notably, the *Architecture Analysis and Design Language* (AADL) [REF-01] – initially developed for avionic applications – is widely used for designing complex real-time embedded systems, mainly thanks to the fact that AADL's language constructs allows modelling both software (SW) and hardware (HW). AADL has a reference implementation called *OSATE [REF-02]*, which is an open-source modelling environment that comes with a few built-in analysis tools, such as flow control and schedulability. Thanks to the fact that OSATE is based on the Eclipse framework, creating new analysis plugins is relatively straightforward. Also, AADL includes an annex mechanism for extending the base grammar, thereby supporting new features and analyses.

AGREE. *Assume Guarantee REasoning Environment* (AGREE) [REF-03] is an example of OSATE annex, which is a compositional assume guarantee-style formal analyses tool. AGREE attempts to prove properties about one layer of the architecture using properties allocated to subcomponents. The composition is performed in terms of formal assume-guarantee contracts that are provided for each component. Assumptions describe the expectations the component has on its inputs and the environment, while guarantees describe bounds on the component's behaviour. The model checker then attempts to find any model execution traces that violate these contracts using one of several Satisfiability Modulo Theories (SMT) solvers. If the model checker covers all reachable states in the model without finding a violation, the model is proven to satisfy its contracts.

Resolute. Another important annex for reasoning over AADL models is Resolute [REF-04], which includes a language for embedding assurance arguments in AADL models and a tool for evaluating the validity of the associated evidence. Because high assurance products generally undergo certification at the system level, there is a natural mapping between a system design and the corresponding assurance argument. Resolute takes advantage of this alignment by enabling the specification of the assurance argument directly in the AADL model. The assurance case can then be instantiated and evaluated with elements specified in the model. The resulting assurance case can be viewed in the modelling environment or exported to graphical tools such as AdvoCATE [REF-05].

Why AADL? The choice of using AADL over other MBE languages such as SysML [REF-06] is informed by multiple factors. First, AADL was designed for specifying hierarchical system architectures, enabling the composition of systems from subsystems, and refinement from abstract to concrete types. It includes first class objects for representing components that comprise embedded systems such as memory, buses, processors, threads, subprograms, and data. SysML, on the other hand, is more abstract and thus better-suited for early stages of system engineering. Second, AADL has sufficiently rigorous runtime semantics, enabling a wide range of analyses that would otherwise not be possible. And third, AADL's annex support cannot be overstated. The ability to extend the language to perform new types of analyses is critical in the rapidly evolving – and heavily regulated – *Cyber-Physical Systems* (CPS) design space.

BriefCASE. On a related note, Cofer et al. recently developed BriefCASE [REF-07], an AADL-based framework for designing, building, and assuring cyber-resilient systems. In that work, high-level security requirements are mapped to seL4 microkernel [REF-08] features via a (very) trustworthy toolchain. Although they did succeed at creating a framework for crafting formally verified secure applications, their work did not focus heavily on hardware security, which plays a fundamental role in protecting a wide range of CPS including IoT devices. In contrast, REWIRE's vision is to propose a framework that is flexible enough to allow system engineers to specify a wide range of system requirements and map them to the appropriate software or hardware block. For example, one might require a platform capable of performing trusted boot to verify the authenticity of an over-the-air firmware update, or a platform capable of executing hardware-implemented crypto-primitives (e.g., symmetric, or asymmetric encryption, hash functions, etc.). While the former security goal could be achieved using a hardware Root of Trust (RoT) acting as the

Trusted Platform Module (TPM), the latter would require Instruction Set Extensions (ISEs) or memory-mapped crypto-accelerators. These solutions are outside the scope of what BriefCASE currently offers.

External Evidence. Furthermore, AADL's Resolute can be used to include formal correctness evidence from tools external to OSATE. This makes it possible for system engineers to choose the appropriate formal technique for verifying each requirement. While some requirements might be better suited for built-in AADL analysis, others might require the use of external model checkers or proof assistants. In REWIRE, for instance, formally verifying a highly undecidable property such as "*crypto-schemes shall be provably secure & resilient to side-channel attacks*" requires a sophisticated tool, such as the Coq proof assistant [REF-09] or external model checkers equipped with sophisticated SMT-solvers.

Formal Verification Techniques. In summary, the two main techniques used to formally verify requirements within REWIRE will be deductive verification with the Coq proof assistant and model checking (besides AADL native built-in analyses).

Coq. Coq is a proof assistant, consisting of a typed high-order functional specification language, a language to write proof scripts, and a proof-checking kernel. It has been granted the *ACM Software System Award in 2013.* Its high praise comes from a variety of reasons: arguably the most significant, Coq relies on a relatively small kernel to check proofs, which means that only a small amount of code must be inherently trusted. Coq also enjoys the strong-normalization property, which informally means that the part of Coq that must be trusted is simple and therefore highly trustworthy [REF-10]. Since its creation, Coq has been successfully used for designing correct-by-construction software and hardware besides formalization of mathematics. Compared to hardware, software development with Coq is mature -- the most notable use of Coq for that goal arguably being CompCert [REF-11], a certified C compiler (the compiler is proved to preserve the semantics of the target program). For hardware design, the use of proof assistants is a more recent endeavour. Researchers at MIT have recently developed a couple of *Domain Specific Languages* (DSLs) for designing circuits and proving its correctness: Kami [REF-12] has been first proposed in 2017 and has since been used to fully verify a RISC-V core implementation. Since then, its development has been picked up by industry. Moreover, Koika [REF-13] is another DSL from MIT that takes the vision set out by the Kami project to a next level: by implementing the core functionalities of the *Bluespec SystemVerilog* (BSV) [REF-14] language, it offers better tools to hardware designers to reason about per-cycle performance.

Deductive verification vs Model Checking. In certain cases, however, highly decidable (i.e., computable) properties are better suited for model checking, since they are optimised for that goal and Coq does not offer a great degree of automation for computing. Coq, and all proof assistants, in general, relies on experts to reason about the design and manually prove properties, whereas model checking works as "push-button" tools, capable of checking properties automatically. As of model checking, in summary, there are two major drawbacks: 1) the limited expressivity of the languages used to specify formulae compared to a tool such as Coq, and 2) the state-space explosion problem, which limits the size of the system that model checkers can handle. There is an inherent trade-off between automation and expressivity when comparing model checking and deductive verification with proof assistants. Thus, REWIRE's vision is to explore both approaches for formally verifying security requirements.

Model Checking. In more depth, model checking is a method for assessing whether a formal model of a system satisfies a given property, expressed in a formal language. Different than dynamic verification, model checking is guaranteed to fully explore the state-space of the design and either validate the properties or generate counterexamples. Historically, after the seminal work that introduced temporal logics in computer science in the 1970s [REF-15], model checking was conceived in the 1980s as an approach to verify computer systems [REF-16][REF-17]. The scalability of the method, challenged by the unmanageable size of the explicit-state models (a problem known as "*state space explosion*") substantially improved thanks to the development of symbolic encodings and algorithms [REF-18], based on efficient data structures [REF-19] and propositional logic. Due also to the advances in the development of satisfiability solvers [REF-20], Bounded Model Checking (BMC) [REF-21] became a successful approach to error finding, later extended to a complete method, both able to detect a violation and prove the validity of a given property, in combination with techniques such as k-induction [REF-22] and Craig interpolation [REF-23]; scalability could be further enhanced by relying on under- or over-approximations,

as in automated abstraction refinement frameworks [REF-24]. More recently, IC3 or property-directed reachability [REF-25][REF-26] established itself as the most effective approach to propositional model checking, as also shown by the results of the Hardware Model Checking Competition [REF-27].

Model Checking & Industry. Model checking, especially in industry, has become the standard choice for verification of digital systems at different levels of abstraction (e.g., RTL, gate-level). Properties of interest are encoded in languages like the *Property Specification Language* (PSL) and *System Verilog Assertions* (SVA) [REF-28][REF-29]. Popular examples of industrial tools for performing property and equivalence checking (specific sorts of model checking) are Cadence's *JasperGold* and Synopsis's *VC Formal*. In the area of hardware security, model checking has been successfully used to detect backdoors and trojans in intellectual property cores and integrated circuits [REF-30][REF-31][REF-32][REF-33][REF-34].

UCLID5. Some works have proposed a mixed approach to the verification of hardware designs. UCLID5 [REF-35], for instance, is a framework that integrates compositional modelling, synthesis, and verification. Given a system model expressed in a proprietary language, the tool supports properties in the form of pre/post conditions, invariants, temporal specifications; depending on the type of property, verification is performed by means of model checking techniques such as BMC and k-induction, with the help of an off-the-shelf Satisfiability Modulo Theory (SMT) solver.

RISC-V. The commonality between all formal verification techniques mentioned above is the fact that they have all been shown to be applicable to RISC-V – one of REWIRE's technological pillars. RISC-V is an **open** *instruction set architecture* **(ISA)** proposed by researchers at UC Berkeley in 2010. Since its creation, it has seen great success, both in academia and in industry. There is a multitude of factors that have contributed to its success, here we cite a few: a) it is royalty free; b) inasmuch as it is open, many parties contribute to its development, which means that innovation happens in a fast and reliable way; c) it is made to be extended, which means that it is relatively easy to introduce new custom instructions; this has been used extensively for tightly coupling cryptographic accelerators to existing RISC-V designs, for instance. Moreover, it is particularly compelling for formal-methods research, because it is a simple clean-slate design to reason about and yet is part of a mature ecosystem. It makes for a very compelling platform for experimentation is basic exploratory research, progressively extensible to full-fledged industrial systems.

Chipyard / Chisel / Rocket. One example of the prolific community effort on the RISC-V development is the Chipyard framework [REF-36]. Chipyard is an open-source generator-based agile design process for hardware, aimed at reducing development costs. Its main features are rich parametrization and incremental extensions. Chipyard provides a framework to bring together a collection of independently developed open-source tools and *Register Transfer Leve*l (RTL) generators. Since its introduction in 2012, Chipyard has been used to generate System on Chips (SoCs) based on the Rocket processor, an in-order RISC-V core. These chips are also called "Rocket Chips" and the corresponding Chipyard feature to generate them is called "Rocket Chip SoC generator". Chipyard inherits Rocket Chip's Chisel-based parametrized hardware generator methodology [REF-37], a Scala-based language for hardware design. Additionally, Chipyard allows IP blocks written on other languages, such as Verilog, to be included via a Chisel wrapper.

SAIL. Finally, the RISC-V Foundation [REF-38] has adopted a golden standard in the form of an ISA specification in the SAIL language [REF-39], meaning that the official RISC-V ISA standard switched from being documented in typical textual form to a formal language which can be linked to a variety of tools, such as Coq. This feature allows designers to reason about correctness directly against the ISA semantics, thus reducing the number of abstraction gaps between different models.

2.2.1.1. Open challenges and needs

Formal verification historically had a strong record in HW verification with a range of different tools and techniques. However, formal verification in crypto-systems and on hardware and software co-design is still unexplored, let alone on safety critical systems. In parallel, formal verification methodologies on RISC-V based security solutions is still in incubation; the research community is currently exploring a series of

established verification approaches close to the hardware level that can reason about functional requirements, usually with respect to memory access (overflows or underflows), mutual exclusion and atomic execution of instructions. A straightforward challenge that the REWIRE project will attempt to tackle is the traceability and provenance of system (device) level requirements down to microarchitecture artefacts that will govern the baseline computations of the processor. At the same time, producing evidence in all the design stages is crucial to validate that each engineer has taken the correct assumptions (verified in next step) to develop his solution. This multi-disciplinary engineering process, enhanced by the REWIRE framework specifically for security requirements will highlight the importance of producing verification evidence per security requirements and when this evidence is requested by the certification authorities. Understanding that security specifications are describing system's non-functional behaviour, the main goal of REWIRE is to instantiate a re-usable approach (e.g., intruder theories and models) that with the appropriate tuning can be applicable to different application use-cases, proving (or not) the validity and resilience of the security solution develop, traced from system level till the instruction set design.

2.2.2. Symmetric cryptography mechanism in the context of physical adversaries

The previous section gave an interesting account of the possibilities and challenges of using formal methods in the design of security systems. However, as the system is deployed, a certain number of additional vulnerabilities are added, such as the possibility of physically accessing or monitoring a device. The notion of security in such a context requires specific countermeasures. Here we describe the context and definitions that enable a provable level of security (i.e., reduction to physical and logical assumptions) to be achieved, taking physical attacks into account.

Authenticated encryption [REF-40]. Often when two parties want to achieve secure communication, they aim at achieving properties that can be defined through cryptography. According to the scenario, they can encompass integrity to ensure that processed data are not forged by a malicious player and confidentiality to assess that the processed data's does not reveal information on the original message. Designing a scheme that combine both integrity and confidentiality is a challenging task and some of the last cryptographic competitions are dedicated to that task. Such constructions that achieve both notions are denoted as "Authenticated Encryption" [REF-41]. They are well studied symmetric cryptographic constructions and therefore offer good security and performance features.

Security of cryptographic protocols and their implementation [REF-42]. A cryptographic primitive can be considered from two points of view: on the one hand, it can be viewed as an abstract mathematical object or black box (i.e. a transformation, possibly parameterized by a key, turning some input into some output); on the other hand, this primitive will in fine have to be implemented in a program that will run on a given processor, in a given environment, and will therefore present specific characteristics [REF-43]. The first point of view is the one of classical cryptanalysis and is somehow encompassed in the previous point; the second one is the one of physical security. Physical attacks on cryptographic devices take advantage of implementation-specific characteristics to recover the secret parameters involved in the computation. They are therefore much less general - since specific to a given implementation but often much more powerful than classical cryptanalysis and are considered very seriously by cryptographic devices manufacturers [REF-44].

Lightweight cryptography [REF-45]. The deployment of small computing devices such as RFID tags, industrial controllers, sensor nodes and smart cards is becoming much more common. The shift from desktop computers to small devices brings a wide range of new security and privacy concerns. In many conventional cryptographic standards, the trade-off between security, performance and resource requirements was optimized for desktop and server environments, and this makes them difficult or impossible to implement in resource-constrained devices. When they can be implemented, their

performance may not be acceptable. Lightweight cryptography is a subfield of cryptography that aims to provide solutions tailored for resource-constrained devices. There has been a significant amount of work done by the academic community related to lightweight cryptography; this includes efficient implementations of conventional cryptography standards, and the design and analysis of new lightweight primitives and protocols. The recent competition of the NIST was completed and “Ascon” [REF-46] was selected as the new standard for lightweight cryptographic application.

Leakage-resilient cryptography [REF-47]. Since the introduction of side-channel attacks in the late nineties, securing cryptographic implementations against leakage has been a major research challenge. These attacks raise critical security concerns, as they enable recovering sensitive information such as long-term secret keys and are virtually applicable to any type of implementation if no countermeasures are deployed. As a result, various types of protection mechanisms have been introduced, working at different abstraction levels [REF-48]. Due to the physical nature of the leakage, the first countermeasures were typically proposed at low abstraction levels. For example, hardware countermeasures can target a reduction of the side channel information by blurring the signal into noise in the time or amplitude domains, or by reducing this signal thanks to special (dual rail) circuit technologies [REF-49]. These hardware countermeasures can then be augmented by implementation-level randomization mechanisms aimed at amplifying the side channel leakage reduction. Masking achieves this goal by exploiting data randomization (i.e., secret sharing) [REF-50] and shuffling does it by randomizing the order of execution of the operations [REF-51]. Steady progresses have been made in order to improve the understanding of these different countermeasures. For example, masking is supported by a strong theoretical background. Yet, it remains that the secure implementation of low-level countermeasures (e.g., masking) is quite sensitive to physical defaults, and is expensive both in software and hardware contexts. At a mode-level, countermeasures can leverage the previously cited protections to mitigate the overall cost of the mode. Nonetheless, a mode-level countermeasures denoted as “leakage-resilient PRF” proposes to use an unprotected primitive in such a manner that makes side-channel attacks hard to achieve.

Formal verification of cryptographic schemes. Formal verification (FV) aims to check if a system meets certain properties by describing the system and the properties mathematically. It is often used for computer-generated/computer-checked proofs. In the present context, FV is used to check (and prove) if the implementation of the AE scheme and its constituents, under certain assumptions, indeed provide security. This can be done in several ways. One way is to check some given (hardware/software) implementation of the AE scheme and its constituents against a high-level formal description (written in some proof-assistant such as Coq) of the AE along with its security properties. Another way is to begin with a high-level description in a proof-assistant, which is known to be correct, and then generate a machine-checked verified C/Verilog code out of it. One interesting goal would be to check if such machine-generated codes indeed provide the required security. We also acknowledge that formal verification tools can be used to verify security properties of masked implementations.

2.2.2.1. Open challenges and needs

Even though several AES modes of operation exist in the literature, to the best of our knowledge they do not formally verify the scheme itself against the corresponding implementation. Given the context above, the target of the REWIRE project is to come up with new symmetric-key operating modes (i.e., AE) protected against passive side-channel attacks. The main innovation in this part is that the side-channel security should not depend on any implementation-level countermeasure [REF-52]. This is an important design goal as it gives freedom to implement the scheme on any IoT platform. Also, the implementation does not expect any special expertise from the engineer. Early research papers in this direction have recently appeared in the literature. However, so far, they were limited to academic proposals and, to the best of our knowledge, have never been realized in a deployment-friendly manner. Thus, there is a need on an AES-based schemes that provide side-channel security by the design implemented in hardware with a parallel architecture (which is available in several IoT-targeted embedded devices). An example of this theoretical construction is called LRBC2 [REF-53] and provides side-channel security by the design

of the AES mode. If certain physical properties can be ensured for this physical building block (which are easily verifiable in a test lab), then we can provably ensure integrity in the presence of leakage. The only resource that needs to be protected in this case is a long-term secret key. Since the long-term secret is used only twice for encrypting a long multi-block message, it has very limited leakage if the AES hardware adheres to the easily testable and achievable physical properties. Other than this physical property, the rest of the leakage-related properties of the mode are attributed to its correctness, and, therefore, can be easily verified by automated formal analysis.

2.2.3. Trust Governance of IoT environments and embedded devices using open standards

A core research pillar of REWIRE is the trust governance of IoTs and embedded devices. This trust governance is based on the underline Root-of-Trust (RoT) such as the Trusted Execution Environments (TEEs). However, existing solutions still lack a precise definition to systematize their design, making each manufacturer offer a different implementation targeting different applications. Also, the adoption of open standards can lead to simpler system designs and could enable more efficient trust management mechanisms.

Intel Software Guard Extension. Intel Software Guard Extension (SGX) is an x86 instruction set extension that offers hardware-based isolation to trusted applications that run in so-called enclaves [REF-54][REF-55][REF-56]. Enclave isolation leverages dedicated hardware-protected memory called Enclave Page Cache (EPC). The OS is responsible for loading the enclave's software in the EPC while the processor keeps track of deployed enclaves and their memory pages in the Enclave Page Cache Map (EPCM). The latter allows the hardware to restrict access to EPC from processes running at higher privilege levels, including the OS or the hypervisor. In particular, the Memory Management Unit (MMU) uses the EPCM to abort any attempt to access the enclave memory that has not been issued by the enclave itself or that does not comply with the specified read/write/execute permissions. Attestation allows the platform to issue publicly verifiable statements of the software configuration of an enclave. In particular, each application enclave has two identities: one called MRENCLAVE formed by the hash of the enclave binary loaded into memory, and the other identity is called MRSIGNER which identifies the enclave developer. During attestation, a designated system enclave – named quoting enclave – outputs a so-called attestation report, i.e., a signature over both identities of the enclave to be attested so to certify that the application runs in an enclave on an SGX-enabled platform. The attestation protocol used by Intel SGX [REF-57] is an adaptation of the so-called Enhanced Privacy ID (EPID) [REF-58]. The latter is a privacy-enhanced version of a group signature scheme that provides some degree of anonymity to the platform where the enclave to be attested is running. In particular, EPID allows two attestation reports on the same enclave to be unlinkable, i.e., no verifier can tell if the two reports were issued by the same quoting enclave on the same platform or by two different quoting enclaves on two different platforms.

SGX also allows enclaves to store encrypted data on disk. This is achieved with a sealing interface that uses hardware-managed cryptographic keys. Sealed data is encrypted and authenticated using keys that are dependent on the platform and on one of the enclave identities. Sealing data against the MRENCLAVE identity ensures that only enclaves loaded with the same binary on the same platform can unseal it; on the other hand, sealing data against the MRSIGNER identity ensures that all enclaves running on the same platform and issued by the same developer (hence, with the same MRSIGNER) can unseal it. Data sealing does not provide freshness, i.e., an enclave that periodically seals its state to disk, has no means to tell if, upon unsealing, the data received is the latest state. This implies that enclaves can be rolled back to previous state [REF-59]. Intel SGX was shown to be vulnerable to side-channel attacks based on cache [REF-60], page-faults [REF-61] as well as so called speculative execution attacks [REF-62].

RISC-V. RISC-V was designed as an open-source Instruction Set Architecture (ISA) with the goal of providing a stable ISA, well-designed and based on well-established principles that could be used

academically or industrially. The original authors of the RISC-V ISA have surrendered their rights to the so-called RISC-V foundation, who allows unrestricted use of the ISA for design of software and hardware. This reduces the cost of innovation and should attract long-term support. To this end, the base ISA and its extensions are developed between the industry, the research, and the educational communities. The ISA is designed to be modular, that is, the ISA base implements a simplified general-purpose processor, and designers can add unrestricted number of extensions to this base. Such extensions may or may not be the ones that have already been ratified by the foundation. This approach allows RISC-V processors to support a wide variety of use cases, e.g., low-power, performance, or compact implementations suitable for embedded systems or high-performance CPUs. According to the Manual [REF-63], RISC-V defines 3 privilege modes, namely Machine Mode (M-Mode), Supervisor Mode (S-Mode) and User Mode (U-Mode) ordered from high to low degree of privileges (Hypervisor Mode (H) will be the 4th mode). The M-mode is the only mandatory mode in a RISC-V processor. Code running in M-Mode is considered inherently trusted, as it has full-access to the hardware.

TEEs in RISC-V. Sanctum [REF-64] was the first designed proposed for RISC-V mimicking the design of SGX but fixing some of its known shortcomings. Keystone [REF-65] developers point out that commercial designs are not only closed but also fixed. This limits the possibilities to add new features or to remove the unused ones to adhere to the small code base principle. Therefore, they designed a fully customizable security monitor and provided a framework to configure, build, and instantiate customized TEEs. CURE [REF-66] allows different kinds of TEEs to adapt to the needs of the applications that require such protection, e.g., it allows kernel-space enclaves, user-space enclaves, or sub-space enclaves. Komodo [REF-67] was a SM originally designed for ARM that was formally verified to make sure that the security properties it was promising hold for every possible scenario. It was later ported to RISC-V and similarly verified. Open-source proposals also fail to adhere to one single standard that would made the tasks of the developers easy and the TEEs compatible between different platforms. Aiming for compatibility and standardization, GlobalPlatform [REF-68] has proposed an API to handle and communicate with TEEs. The Open-TEE project [REF-69], presents an open-source implementation. Although it was originally designed for ARM TrustZone, it is designed so it can be compatible with any underlying technology. Therefore, Keystone [REF-65] could be extended and adapted to adopt the GlobalPlatform specifications.

The TEE designs based on RISC-V use the different security primitives available in the specifications, and therefore in the processors, to achieve some of the goals of the TEE e.g., isolation. Namely, RISC-V based TEEs are built relying on Physical Memory Protection (PMP) registers and on the aforementioned privilege modes. The Security Monitor running in Machine mode will take care of handling the PMPs and will manage context switches from and to the TEE. The PMP unit provides per-hart machine-mode control registers to allow physical memory access privileges (read, write, execute) to be specified for each physical memory region [REF-63]. The size of such regions is platform-specific, but the standard PMP encoding supports regions as small as four bytes. PMP checks are applied to all memory accesses whose effective privilege mode is S or U, including instruction fetches. For TEE designs, the PMPs are configured so the enclave memory can only be accessed by the enclave itself and the SM, which is considered trusted as it manages the TEE. Additionally, RISC-V defines an Input/Output Memory Management Unit (IOMMU) that is used to regulate the access to main memory by peripheral devices in a computer system. Without IOMMU, malicious peripheral devices can directly access memory through Direct Memory Access (DMA). IOMMU introduces the necessary data structures for mapping peripheral devices to individual address spaces while maintaining compatibility with existing page table formats. Further, it is designed to be flexible, scalable and to allow integration with virtualization-based systems.

Keystone's Security Monitor exclusively uses standard RISC-V features, making it easy to port to different hardware platforms able to work on RISC-V. This can also serve a purpose of defending against certain attacks, without any application changes, by only customizing the Security Monitor (SM) for a specific platform so that it can defend the enclave against a physical attacker or cache side-channel attacks [REF-70]. The SM is trusted machine mode software that does not perform any resource management, only determines, and enforces security boundaries on physical memory regions and acts as the Trusted

Computing Base (TCB) for the system. There are several benefits for using an M-mode software as the TCB [REF-71]:

- Programmability: Unlike microcode, we can build M-mode software in existing programming languages (i.e., C) and toolchains (i.e., gcc).
- Agile Patching: Since the SM is entirely software, it is possible to patch bugs or vulnerabilities without involving hardware-specific updates.
- Verifiability: In general, software is easier to formally verify than hardware.

The SM can also be extended with further features if needed. TEEs are becoming a commodity in a number of proprietary systems, but Keystone is the first open-source framework for building TEEs. Not only the capabilities of it, but also the recentness of Keystone (some introductory papers date from only 2020) make it even more interesting from the perspective of innovation in REWIRE.

2.2.3.1. Open challenges and needs

All REWIRE use cases are based on scenarios where low battery consumption and smaller footprints are preferred. The adoption of open standards (e.g., RISC-V and Keystone) has revealed new opportunities for trust governance of IoT environments and embedded devices, yet several challenges need to be addressed by the REWIRE project. For instance, while the resource constraint embedded devices of IoT environments mandate the adoption of lightweight TEE solutions, Keystone presents itself with limited documentation and many in-progress features.

Also, until now many solutions for IoT environments were based on TrustZone. However, that vendor-locked option only enables one big, isolated area, making it very hard to establish multiple isolation domains [REF-72]. In addition, in commercial solutions each vendor rolls out its own TEE implementation with a particular design for their chosen threat model, and consequently present limited design options for adopters, who are required to make an effort to port their application to run in their target TEE, while these commercial solutions most times are not compatible with other architectures. Further, closed designs and scarce documentation limit the capabilities of researchers to analyse their security properties. Therefore, with the development of RISC-V, a number of proposals to build TEEs based on it appeared. Moreover, very diverse technologies must be integrated for the different use cases; thus, working with RISC-V based hardware seems optimal and a guarantee that future developments could be added to the platform more easily (openly), which adds to the sustainability and exploitation dimensions of REWIRE.

2.2.4. Runtime monitoring of IoT trustworthiness and operational assurance of Systems-of-Systems

A core research direction of REWIRE is the efficient runtime monitoring and the operational assurance in resource constrained Systems-of-Systems (SoS) based on novel attestation schemes.

Remote Attestation. Attestation is the process by which one party, the 'Verifier', assesses the trustworthiness of a potentially untrusted peer, the 'Prover'. In all attestation methods, evidence is gathered and signed by a more trustworthy attester [REF-73]. Remote attestation was first introduced by the Trusted Computing Group (TCG) during the first decade of the 21st century. TCG specified a Trusted Platform Module (TPM-based solution for verifying the identities of loaded boot images [REF-74]. This was followed by key attestation for smartphones, verifying that a credential is protected by the device's TEE [REF-75]. In recent years, there has been a large interest in various remote attestation protocols, including swarm attestation [REF-76], anonymous attestation [REF-77], and mutual attestation [REF-78].

Three main remote attestation approaches proposed in the literature (a) the software-based, (b) the hardware-based, and (c) the hybrid, based on the underlying RoT. Software-based remote attestation schemes do not require a hardware RoT. Software-based remote attestation constitutes of self-measurement mechanisms implemented entirely in software and/or formally verified. Even though software-based solutions can reduce cost and be widely adopted, most of them build upon strong

assumptions. Hardware-based techniques perform remote attestation based on a dedicated hardware component, e.g., TPM. Hardware-based attestation can provide higher security than software-based ones since the tamper-resistant hardware ensures the correctness of the attestation results. However, hardware-based solutions are inapplicable to devices (e.g., IoT) that are not provided with the customized hardware components in the production phase. Hybrid attestation schemes leverage the advantages of both hardware-based and software-based schemes and rely partially on dedicated hardware.

Configuration Integrity Verification. Configuration Integrity Verification (CIV) focuses on the correct configuration state of a platform (e.g., configurations or binaries) that it has not been compromised by a malicious party, ensuring the trustworthiness of the device configuration [REF-79]. However, static remote attestation, such as CIV, provides no information about the order in which instructions within the binary execute during runtime. More specifically, control flow hijacks [REF-80], code reuse attacks [REF-81] or return-oriented programming [REF-82] can change the order in which instructions execute without modifying the binary. These types of attacks are considered the most devastating since they try to exploit memory- and data-related vulnerabilities for altering the execution path of the underlying system processes and cannot be detected by static remote attestation techniques, since the binary remains the same.

Control-flow Attestation. To address this limitation, Control Flow Attestation (CFA) [REF-83][REF-84] augments static remote attestation to provide to the Verifier with an unforgeable control flow proof containing the order in which the instructions of the attested binary have executed. As such, CFA enables the detection of control flow hijacks and code reuse attacks, even when these attacks do not modify the installed binary. In other words, CFA is used to verify that the execution flow or a Control Flow Graph (CFG) of a software process running on a device is benign and has not been compromised or influenced by an attacker.

Tracing. One of the core building blocks in any attestation scheme is the underline tracing mechanism capable of extracting the necessary system measurements to be used for verification. Several open-source tracers exist in the literature such as the Unix-based ftrace tool [REF-85] that provides static and dynamic tracing and SystemTap that provides dynamic tracing based on Kprobes, Jprobes, and Uprobes [REF-86]. However, both ftrace and SystemTap are TRAP-based methods, meaning that they consume a significant number of resources, which is infeasible for resource-constrained devices. DTrace [REF-87] is another dynamic tracing mechanism, that offers very limited optimizations, which limits its applicability too. Linux Trace Toolkit Next Generation (LTTng) [REF-88] is another tracing tool that adds up considerably the collective tracing impact on the target software for long runs in resource-constrained devices [REF-89]. The main tracing approaches in the literature are a) software tracing, b) hardware tracing and c) hardware-assisted software tracing.

2.2.4.1. Open challenges and needs

Several challenges need to be addressed in the context of REWIRE project. For instance, an open issue is how to address the non-efficient attestation procedure, especially the CFA in resource constrained devices is infeasible. Thus, REWIRE needs to investigate more efficient attestation schemes that consider only the minimum number of properties to be verified minimising the produced overhead, in combination to the formal verification of software and hardware co-design. On top of that, a research question that is erased is how to extend the functionalities of the underline TEE (e.g., keystone) to support the new bread of attestation schemes. In parallel, in all tracing approaches, there is always a compromise between performance, security, and usability. This sets the challenge ahead on how to provide tracing capabilities capturing the requirements of REWIRE remote attestation while striking a balance between the precision of monitored system traces, and efficiency. Last but not least, an open question that might be considered in the context of REWIRE is how to capture the sequence of ISA commands in the context of RISC-V or in another research direction is how to use monitoring hooks.

2.2.5. Decentralized identity management and trust-aware continuous authentication and authorization

The basis of decentralized identify management is the recent developments in Self-Sovereign Identity (SSI). SSI is an umbrella term for an identity management system and is a digital identity that is in the control of the holder and does not depend on a centralised registry, Certificate Authority (CA), or identity provider. The main building blocks for SSIs are decentralised identifiers and verifiable credentials. The authors of [REF-90] describe the architectural stack of a typical SSI system. Many SSI systems (e.g., Hyperledger Aries) implement this stack, which suggests that DIDs, blockchain and verifiable credentials/presentations are necessary for an SSI system.

Decentralised Identifiers. Decentralised Identifiers (DIDs) are unique identifiers that in a similar way to URLs resolve to a DID document that contains the data necessary for the holder to prove that they own that identity. At its simplest, the holder (usually the owner of the identity-related information) will generate an asymmetric key pair and the public key will be included in the DID document together with details of the algorithm being used. When challenged to confirm ownership the holder can use the private key to sign the challenge, and this can be verified by the challenger using the given public key. DIDs are being standardised by the World Wide Web Consortium (W3C) [REF-91]. Additionally, the DIDComm Messaging specification aims to provide a secure, private communication methodology built atop the decentralized design of DIDs [REF-92].

The DID documents need to be stored on a data registry made generally available so that the DIDs can be resolved. This data registry could be a permission-based web domain or a distributed ledger (either permissioned or permissionless). There are benefits and drawbacks to each choice [REF-93]. However, permissioned or permissionless, the distributed ledger has a clear advantage in that it is tamper-proof. In the context of SSI, the authors' of [REF-93] provide an evaluation of whether *trust* can be obtained from a distributed ledger such as a Blockchain (BC) infrastructure. In the permissionless setting it is found that trust requirements are not met since there is no guarantee that the issuer is who they claim to be, or that the issuer will behave honestly when issuing credentials. In permissioned BC, the burden on users' and verifiers is reduced since an administrator is introduced to check the consistency of transactions and control what is logged on the ledger – this means a reliance of trust in the administrator.

Verifiable Credentials. Verifiable Credentials (VC) are statements that the holder has a given credential or satisfies some specified condition. In general, the credential will be provided by an issuer who confirms the identity of the holder and checks that they also have the necessary credential (e.g., a driving licence). The generated VC, to be verified, must contain details of the holder, the DID of the issuer, information about how the credential being confirmed and details of how the issuer signed the information in the VC (e.g., which crypto algorithm was used). Both the holder's DID, and any VCs are held in a *digital wallet* for security purposes. This wallet can be in an edge environment or cloud environment, depending on the application.

Verifiable Presentations. The holder may have many VCs and can use all, or a selection of them, to create a Verifiable presentation (VP). What information is included in a VP is under the control of the holder. Both VPs and VCs are standardised by the W3C [REF-94]. In a similar way to the use of OAuth and Jason Web Tokens (JWT), a VP can be used for trust-aware authorisation. The key used to generate the VP is under the control of the digital wallet, while access policies are used to restrict its use and therefore control whether the device can get access, or not. In general, the use of the key can depend on the assessed trustworthiness of the device. In [REF-95] an identity management system is proposed using W3C verifiable credentials and Fast IDentity Online (FIDO). Whilst it is not called an SSI system, it shares some properties of SSI systems, e.g., decentralisation of identity management. This paper provides more detail regarding the necessity of DIDs for SSI.

DIDs, VCs and VPs use proofs (e.g., signatures on their data). The current mechanisms used for these proofs is described in the W3C documents relevant for data integrity and for proof of ownership or authorship [REF-96][REF-97]. In addition, proofs that allow presentations to confirm ownership of certain credentials while keeping other information private, are of particular interest. Known examples exploring

these technologies are:

- The European Blockchain Partnership (EBP). The EBP was set up in 2018 by the EU to use distributed ledger technology to enable public administrations, businesses, and individuals to create trustworthy Europe-wide services. To do this, the EBP defined a specification and made provision for the European Blockchain Services Infrastructure (EBSI) [REF-98]. This infrastructure is in place and currently being used by 20 projects that have joined the early adopters' scheme. The system allows a range of digital wallets to be used for storing the credentials.
- Microsoft Azure Active Directory Verified Credentials [REF-99]. This system can use a distributed ledger (Bitcoin or Ethereum, for example) or a permission-based web domain. In this case the Microsoft Authenticator App is used to store the credentials.

While the discussion above has talked of holders and referred to physical entities like driving licenses these protocols and procedures can be used for any device with a unique identity and has a digital wallet to securely hold the keys and other necessary data.

Anonymous Credentials. The AnonCreds working group is currently trying to produce specifications for *Anonymous Credentials* (AC) which are the primary standard for Zero-Knowledge Proof (ZKP)-based verifiable credentials. The v1.0 specification is based on an open-source verifiable credential implementation of AnonCreds that has been in use since 2017, initially as part of the Hyperledger Indy open-source project and now in the Hyperledger AnonCreds project [REF-100]. An AC [REF-101] can be thought of as a *special case* of VCs. To see this, we make the following observations:

W3C VCs have the holder (sometimes called the prover) binding process occur without any additional interactions between the holder and the issuer, thus, there is a lack of holder privacy. The AnonCreds specification counters this by requiring that the holders are bound to the AC by a non-correlatable secret only known to the holder. This secret is called a *link secret*, which can be viewed as a *blind attribute*. This alternative to persistent identifiers means that *selective disclosure* of attributes is possible. Observe that most AC schemes encode more than one attribute and a key goal of research in this space is *minimal disclosure*. For practicality and real-world implementation, *pseudonyms* are a useful tool in AC schemes [REF-101]. Access to a pseudonym means attaching a consistent *alias* to *each* credential. This alias can then be used to maintain a registry of which credentials have been *revoked*.

One piece of terminology that needs to be distinguished is the use of the term "proof presentation" in the W3C VC and AnonCreds specifications [REF-94][REF-100]. This term describes the holder presenting one of three items to the verifier: (a) all the data in the credential, (b) part of the data in the credential (a partial disclosure) and (c) a ZKP for some data in their credential. Using ZKPs is a critical aspect of Hyperledger AnonCreds [REF-100]. Explicitly, it is not the actual VC that is presented to the verifier, but rather a cryptographically derived *proof* that allows the data in the credential to be presented securely. Therefore, the user can retain and control privacy whilst responding to questions like "Are you in the age category of 25-30?" Alternatively, the W3C standard for the Verifiable Credentials provides the same basic functionality as AnonCreds, such as supporting ZKPs, but not all W3C VCs support being asserted as ZKP responses. The choice of whether to enable this capability is left up to the credential issuer, therefore, VCs as per the standard are not viewed as AnonCreds. For a holder to assert a ZKP response in the VC data model, they must use a credential that has been created to allow for this purpose.

Identity Wallets. Privacy preservation has led to the concept of user-centric identity management, using identity wallets, applied to the DID domain to support the SSI paradigm. Typically, a wallet stores and manages cryptographic material and verifiable credentials/presentations related to identity-based attributes. Applying digital wallets to the DID domain is useful to avoid digital identity-related data being stored and managed by a central authority. Moreover, improving the security and privacy guarantees of identity-related information is desirable, as is user control of their identity-related information. The authors of [REF-102] have recently provided a systematic review of many works related to digital wallets to contextualize the terms and concepts used across recent literature. Essentially, a DID wallet is software operating in a remote (cloud) or local (edge) environment (sometime referred to as an "Agent") for the purposes of identity-related data storage, management, and sharing. Given such a wallet, a user controls

and manages their data stored in the wallet - including removal/reviews of the identity-related information. Moreover, the user can decide when and where (in or out of the wallet) the data is shared and stored. Further features include allowing for the combination of identity-related data when the data is outside of the wallet, upon the users' consent. The underlying environment should also support the recovery and back-up of a users' identity-related information.

In the context of wallet implementation, [REF-103] recently delved into the issues surrounding trust and DID wallets (recall [REF-93]). The authors' work addresses this issue by providing a systematic analysis of a selection of available digital identity wallets on the market in terms of the technologies used to establish trust and controlled sharing of data. In the context of regulation, federated identity-management system eIDAS is working to standardize cross-border authentication processes across EU member states by utilizing wallets in the DID domain to support SSI. However, there are challenges facing this process as discussed in [REF-104]. In particular, the contradictions of SSI and eIDAS requirements related to trust, privacy and data-sovereignty. It is noted by [REF-104] that Distributed Ledger Technology (DLT) is not necessary to support SSI. The pervasiveness of data-stored using DLT contradicts requirements like privacy-preservation and features such as erasure of identity-related information [REF-93]. Therefore, [REF-104] concludes the restriction of DLT to meet a sufficient level of trustworthiness, privacy, and data-sovereignty. The authors' of [REF-105] provide a review of the top five mobile cryptocurrency wallet. Revelations from [REF-105] ought to be taken into consideration, especially with regards to the *users' experience*. Specifically, shortcomings such as a users' misconceptions, some of which can be traced back to a reliance on their understanding of conventional centralized systems. The result is that users are often presented with unmet expectations. Based on the findings of [REF-105], the authors provide recommendations on how to design cryptocurrency wallets that both alleviate issues and counteract some misconceptions to better support onboarding customers.

TC-based Identity Wallets. Trusted Component-(TC) based identity wallets can be viewed as more secure than SW wallets, however, their capabilities, such as accessibility, can be constrained [REF-106]. The authors of [REF-106] compare HW and digital crypto wallets, concluding that both wallets are very secure in the right scenario and with appropriate use. In [REF-107], different approaches to implementing digital wallets using TPM-based functionalities are studied. Traditionally, a TPM is HW utilized for PKE, hashing, MACs, Direct Anonymous Attestation (DAA), key migration, and data migration. In the context of digital wallets, TPM-based *monotonic counters* can be used as a building block for implementation. Informally, a virtual monotonic counter is a mechanism (hardware/software) that stores a value and provides the {read, increment} commands for access to the value. In conclusion, [REF-107] determined that a hash-tree mechanism achieves lower performance overheads compared to log-based ones. However, the current TPM specification does not support the former. Therefore, current mechanisms for TPM-based wallets have relatively high overheads.

2.2.5.1. Open challenges and needs

The most prevalent challenge with respect to DID management within the SSI paradigm will be *trust-establishment* of credential issuers to ensure trust-aware continuous authentication and authorisation. This issue is highlighted in [REF-108], which focuses on balancing *trust* and *governance* requirements in a decentralised, privacy-friendly identity management system. As a direct consequence, REWIRE will need to propose a **novel**, practical, and scalable mechanism to support verifiers to externally verify their trust in credential issuers. Following the approach taken in [REF-108], this could be achieved by creating a trust registry component within the SSI ecosystem such that elements in the registry, coupled with additional information, will support a verifier in making informed trust decisions. Moreover, REWIRE needs to consider the appropriate use of DLT to meet a sufficient level of trustworthiness, privacy, and data-sovereignty. Recall the debate regarding federated identity management systems and SSI ecosystems presented in [REF-93][REF-104]. DLT is not necessary to support SSI, however, given its use in REWIRE, the revocation or renewal of identity-related data will be a challenging task. This is due to the persistence of stored data which contradicts properties like privacy-preservation. Moreover, ensuring the compatibility of digital wallets and VCs with the IoT ecosystem in REWIRE will also need to be carefully considered whilst additionally improving efficiency and costs of handling large amounts of data.

2.2.6. Firmware and software automated validation and vulnerability analysis

Various tools and platforms exist for the unpacking of firmware. The most accurate and widely applicable ones are Binwalk [REF-109] and BANG [REF-110]. These tools do not serve as specialised unpackers for firmware per se but can carve information from any blob of data. For example, these tools can recognise the used headers and unpack a compressed archive. In cases where no headers exist, they attempt to recognise a structure for identifying a method (e.g., a file system, a vendor-specific method of packaging firmware, a compressed file system, or a file type like an image) of storing information and finally they attempt to extract the information using this method. The firmware validation of the REWIRE project takes complete firmware images as input. This means various formats of packing said firmware must be supported, as different vendors tend to use different ways of packing their firmware. Moreover, unpacked firmware only contains the compiled version of the programs it uses, meaning the validation must support bytecode analysis. In general, four main approaches exist in the literature to analyse the firmware.

Static Analysis. It can be analysed statically, where none of the firmware is run. The advantage of this approach is that it is always usable, and there are no dependencies on emulation or hardware. The disadvantage is that most firmware has interactions that are too complex to analyse statically.

Dynamic Analysis. Second, firmware can be analysed dynamically by selecting individual components, such as an HTTP server, and emulating this. The emulation can then be analysed using techniques such as fuzzing, concolic execution, and taint analysis. Both its advantage and disadvantage are the targeting. It can effectively 'attack' single components and find flaws more quickly than a broad analysis. When not targeting the right attack surface it is prone to missing important flaws, however.

Emulation. The firmware can also be emulated as a whole. The advantage of this approach is that the most accurate picture can be formed out of all emulation-based approaches, but the difficulty of achieving this is a disadvantage.

Analysis on the hardware. Lastly, firmware can be analysed on the hardware was built for. Its obvious advantage is the guaranteed accuracy of the analysis. It does not scale well however, and it can be more difficult since there is relatively little control over the inner workings of any given device.

In practice a combination of the above approaches is used, as they can complement each other's strengths and weaknesses. There are many ways to **statically analyse firmware**. Not all are included in this project, but a selection of the most widely used ones are included below:

Static taint analysis. In taint analysis user input is marked as a ‘tainted’ source. Each action a program takes can spread this taint, such as copying information or using it as input for a calculation. The analysis can specify certain ‘sinks’, which are points in the program that should not be reachable by tainted data. If this happens anyway, the path through the program which allows this to happen is marked as suspect. Taint can be cleared from data, for example through input validation. In source code this technique can be applied more easily than in compiled programs, as the compiled program would need to be interpreted by something that is aware of the workings of the processor it was compiled for.

Static symbolic execution. Symbolic execution parameterises user inputs. At every branch a program takes a constraint is tracked, where each path through the program will result in its own unique set of constraints. The constraints to each such set can be fed to an SMT solver, which can solve them and thus tell which inputs are required to follow this path through the program. In static symbolic execution a compiled program must first be disassembled and lifted to an intermediate language, which can then be interpreted by the symbolic execution engine. A downside to the technique is the path explosion problem, where the engine has to keep track of so many different paths and constraints that it runs out of memory. For example, given a loop in a program that loops one hundred times with a branching condition inside, 2^100 paths would have to be tracked.

Control flow graph analysis. This technique can be used to fingerprint the behaviour of a program. In disassembled code it checks which basic blocks are connected to each other by control flow transfer instructions, such as jumps. A graph is created where each basic block is a vertex and each transfer to another basic block is an edge. This graph can be used for further analysis, or mitigation techniques such as control flow integrity protection.

Like static analysis methods, there are many **dynamic ways of analysing firmware**. The most prominent ones are listed below:

Dynamic taint analysis. Just as in static taint analysis all user inputs are tainted. However, in this case it is done while running the program that is being analysed. In practice this means that each register, as well as the stack and heap, have a twin that tracks whether the data it holds was influenced by user input. An advantage over static taint analysis is that many more unconventional inputs can be tried, which can result in unexpected behaviour that spreads taint further.

Concolic execution. In concolic execution the program is executed as normal, using real user input. However, at every branch it also keeps track of constraints required by both paths. Multiple variants exist here, where both the path taken by the concrete execution and the path not taken are evaluated, or where only the concrete path is evaluated. The first approach still suffers from the path explosion problem, whereas the second approach provides less coverage.

Fuzzing. In classic fuzzing, random input is sent to a program until it crashes. After it crashes the input which caused the crash is analysed to see why it made the program crash. In this manner unexpected behaviour of the program can quickly and easily be identified. In more recent years this technique has evolved to use more information from the program itself. The most well-known fuzzer is AFL++ [REF-111], which uses coverage guided fuzzing. For each branch in a program a small snippet is inserted that signals “this path was reached” to the fuzzer. Inserting these snippets is called instrumentation, which has its own challenges for binary code and thus is described further down below. The fuzzer can use this information to determine when an input was interesting, i.e., reached a new path. Any interesting inputs are reused and mutated, whereas uninteresting inputs are discarded.

Binary instrumentation can be approached in two different ways.

Logical Approach. The first is the logical approach of disassembling the program, inserting the instrumentation code, patching internal references such as jump addresses, and reassembling the program.

Sub-instructions. The second approach makes use of so-called sub-instructions, which requires an emulation platform to be used. This cannot alter the program itself, but allows specific instructions to be instrumented, or even specific addresses. The first approach is generally faster, since it could be run as native code. However, patching the internal references is a highly complex issue and sensitive to mistakes. The generally used approach is therefore the second one.

In the context of REWIRE a solution that combines fuzzing and concolic execution will be adopted. A wide variety of techniques exist under the umbrella of fuzzing, which cannot all be covered here. In parallel, for the emulation of OS-level functions the Qiling project [REF-112] will be used, which in turn uses Unicorn [REF-113] for CPU emulation, Capstone [REF-114] for disassembly, and Keystone [REF-115] for assembling. Also, AFL++ will be used as a fuzzer, with angr [REF-116] as the symbolic execution engine.

Monitoring Hooks. Attestation of the correct working of runtime processes requires constant information to be collected from these processes. This will be implemented by instrumenting these processes (e.g., using monitoring hooks on the processes) and having them report information to an agent running in the TEE. Since no access to the source code of the firmware is assumed, and monitoring should take place during runtime on embedded devices, REWIRE will make use of static binary instrumentation. Conventional solutions addressing the need for control flow integrity [REF-117] make use of dynamic binary instrumentation, which allows for a more general solution in these protections. However, these

solutions need to run together with the process they instrument, for example by adapting the loader used to run the process. Another existing type of solution is through the use of the control flow integrity extension of the RISC-V instruction set, Zisslpcfi [REF-118]. This solution is not yet usable in the REWIRE project since it is a tentative specification and would require hardware support for this specific extension, as well as adoption by developers. The need for runtime monitoring such as control flow integrity is highlighted by recent research showing the RISC-V architecture, like most other architectures, is susceptible to control flow hijacking [REF-119]. Existing solutions implementing static binary analysis aim for security and speed [REF-120], applicability for deeply embedded devices [REF-121], and a low memory footprint [REF-122]. Since the instrumentation will be used to implement control flow integrity to arbitrary firmware, the REWIRE project will focus on security and speed.

In parallel with SECURA, Eight Bells conducts research and development efforts focused on firmware deployment and vulnerability analysis. Our work on researching additional tools for these purposes while also delving into existing literature to explore new methods and new vulnerabilities. The overarching objective is to help maximize the effectiveness of firmware deployment and analyzation efforts.

For our first steps we explore tools such as:

ISA Detect is a software utility employed in software development and system optimization processes to determine the specific instruction set architecture (ISA) supported by a Central Processing Unit (CPU) or microprocessor. This information is vital for developers as it guides decisions on how to compile code, optimize software performance, and ensure compatibility with the target hardware. The tool typically functions by examining the CPU's internal registers and hardware-specific characteristics to identify its ISA. Armed with this knowledge, developers can make informed choices about compiler settings, code optimizations, and software configurations to ensure efficient and effective execution on the target CPU architecture.

Centrifuge: uses two new approaches to analysis of file data: DBSCAN, an unsupervised machine learning algorithm, is utilized to uncover clusters of byte sequences grounded in their statistical characteristics, or features. Byte sequences that encode identical data types, such as machine code, typically exhibit similar properties, making clusters a reliable representation of a specific data type. These clusters can be subsequently extracted and subjected to further analysis. To pinpoint the specific data type within a cluster, one can often forgo machine learning and instead rely on the calculation of the **Wasserstein distance** between its byte value distribution and a reference distribution corresponding to a particular data type. If this distance falls below a predefined threshold for that data type, the cluster is confidently identified as that particular data type. Currently, reference distributions are available for high entropy data, UTF-8 English, and machine code tailored to various CPU architectures. The Wasserstein distances between the data in the clusters and the reference distributions are assessed to determine the type of data in the clusters after DBSCAN locates them. Centrifuge utilizes ISAdetect to determine the target CPU of any machine code it finds in the file.

2.2.6.1. Open challenges and needs

While advances in the automation of binary analysis have uncovered many flaws in software, fully automated software testing meets several problems. In most fuzzing and symbolic execution solutions a so-called ‘harness’ is created, which is a program that encapsulates the targeted program. By doing so it can, for example, target specific interesting functions of the target program, such as a parser. When aiming to automate such testing for general cases, the software needs to undergo preliminary analysis before testing it using these techniques. This preliminary testing should yield information such as which input vectors the program uses, what input formats these use, and what kind of protocols are used by the program. Only then could an effective harness be created automatically. State of the art provides little towards this need, which is a challenge that needs solving in the REWIRE project. Without doing so, the automated analysis would not be usable on all software, and where usable would usually not be fast enough to be feasible.

A second challenge within the aforementioned challenge comes with the specific use cases in REWIRE, which are all embedded systems. Since such systems generally rely more heavily on hardware-specific

implementations, scaling automated analysis requires either control over said hardware or emulating the software. For the sake of scalability, the REWIRE project opts for emulation. Embedded systems tend to rely on peripherals for their functioning, which in some cases are crucial for the correct functioning of the software. Automatically detecting such peripherals based on the software is a challenge and influences the accuracy of a testing harness as well. Last but not least, another challenge is how to model these monitoring hooks (e.g., binary instrumentation) for the purpose of formal verification. This is crucial for verifying that the security and correctness of the binary is not tampered with or violated by the instrumentation process.

2.2.7. Service certification and auditing through blockchain-based secure information and data exchange

Blockchain technology can be used to provide a secure and decentralized platform for service certification and auditing. With the use of blockchain, service providers can store and exchange their data and information securely, transparently, and immutably.

Blockchain and Oracles (capabilities and data management). Blockchain technology has recently emerged as one of the promising solutions for data management systems. In general, blockchain is a decentralized ledger technology with a “chained blocks" data structure; every block header includes the root of a Merkle tree, a timestamp, and a hash of the previous block [REF-123]. When data inside the previous block is changed, the new hash is never referenced to another block, resulting in rejection by the network. Consequently, once data is committed to the chain, it is immutable, decentralized, traceable, and verifiable. Combining these properties with a data management tool can achieve the features, e.g., data integrity, increased security, and traceability to achieve better data management. The potential of blockchain as a data management system has been investigated in recent study [REF-124].

On the other hand, blockchains cannot pull in data from or push data out to any external system as built-in functionality. This is known as the blockchain oracle issue, which highlights the inability of smart contracts to confirm the veracity of external data [REF-125]. To handle this problem, traditional blockchain-based systems usually use oracle services to record events that occur outside the blockchain. An oracle service can be regarded as a secure middleware to facilitate data communications between the blockchain and any off-chain systems [REF-126]. Using oracles in data management systems fills this gap and ensures that the real-world data fed into the blockchain is accurate and the smart contract is triggered properly.

Secure Oracles. Traditional centralized blockchain oracles are efficient but susceptible to targeted attacks, e.g., a single point of failure. Decentralized oracles, although they avoid the issues, are often inefficient [REF-123]. In this respect, researchers are trying to build decentralized oracles using different mechanisms, but there is still room for improvement in terms of trustworthiness, scalability, efficiency, and privacy [REF-124]. To address these challenges, trusted hardware (e.g., Intel SGX or ARM TrustZone) could be combined with decentralized oracles to create such a "secure oracle" protocol. For example, TEEs can help address inaccuracies or inconsistencies in the external data source by enabling the oracle to filter out unwanted or malicious data before it is transmitted to the blockchain [REF-125]. In this way, trusted hardware could increase the trustworthiness of data filtering and collation, as well as the authentication of data sources, thus improving the efficiency of blockchain oracles and providing a secure and private data processing/filtering environment. Further research is needed in REWIRE to explore the full potential of this approach for real-world blockchain applications.

On-off chain data management. In general, on-chain and off-chain data management are two different approaches to managing data in a blockchain network. Both on-chain and off-chain data management have their advantages and disadvantages. On-chain data management provides high transparency and immutability since all data is stored on the blockchain and can be easily audited. However, keeping large amounts of data on the blockchain can be costly and slow down the network's performance. Off-chain data management, on the other hand, can be more cost-effective and allow for faster processing and is therefore more popular. The recent state of the art in on-off chain data management for blockchain technology includes various solutions such as sidechains [REF-127], sharding [REF-128], and state

channels [REF-129]. Sidechains and state channels provide off-chain storage solutions while still maintaining the benefits of the main blockchain, while sharding allows for better network performance and scalability. The ongoing research in this area is focused on finding new ways to balance the benefits of on-chain and off-chain data management in the context of blockchain technology.

Smart contracts for data collection. Blockchain was originally introduced as the underlying technology for Bitcoin. Now, with smart contract technology bringing powerful programmability, it is widely believed that blockchain can be applied to build decentralized systems in various application scenarios, e.g., healthcare, finance, energy trading, wireless communication, service allocation, electronic voting, and supply chain management [REF-130]. A smart contract is a tamper-proof and self-executing program running on the blockchain, which enables a much broader range of application innovations in addition to cryptocurrencies. The concept of smart contracts has also extended to other blockchain platforms, e.g., chaincodes [REF-131] and transaction processors [REF-132] are smart contracts offered by Hyperledger Fabric and Sawtooth, respectively. Smart contracts can be a powerful tool for automating data collection processes in a secure and efficient manner, which in many cases requires the help of secure oracles.

Blockchain wallets and verifiable credentials. Blockchain wallets are digital wallets that are used to store and manage cryptocurrencies, such as Bitcoin and Ethereum. These wallets are built on top of blockchain networks and provide a secure and tamper-proof way of managing digital keys and assets. In this respect, TEEvault [REF-133] is a hardware-based key management solution for cryptocurrencies that securely stores and manages keys in an enterprise environment. The entire process of key generation, backup, and usage is completed inside a protected TEE, safeguarding keys from hacking or physical attacks. Verifiable credentials, on the other hand, are digital credentials that are based on blockchain and allow individuals to prove their identity, qualifications, and other attributes in a secure and tamper-proof manner. The state of the art on this topic involves the development of decentralized, tamper-proof systems for issuing and verifying digital credentials using blockchain technology. Some of the key developments include the use of DIDs to provide a decentralized and secure way of storing and sharing verifiable credentials [REF-134].

Access control on blockchain. Since blockchain is a decentralized and distributed system, access control mechanisms are essential to ensure the security and integrity of the network. By implementing access control mechanisms such as public and private blockchains, cryptographic keys, role-based access control, and smart contract-based access control, blockchain networks can be configured to provide access only to authorized users [REF-135]. This reduces the risk of unauthorized access and tampering, thereby enhancing the trustworthiness of blockchain-based transactions. Some permissioned blockchains also involve special access control mechanisms. For example, in the Hyperledger Fabric framework, channel technology is used to create private communication channels between specific network participants [REF-136]. As blockchain technology continues to gain adoption in various industries, effective access control mechanisms will be crucial to enabling secure and reliable interactions on the network.

Blockchain on certification and auditing. Data integrity ensures the correctness and consistency of data throughout its life cycle. It, therefore, plays a vital role in the design, implementation, and utilization of any data management system. The current solution for data auditing and certification is through third-party auditing service providers such as Spectra [REF-137] . However, these centralized services are not immune to malicious auditors and therefore suffer from a Single Point of Failure (SPOF). Besides, the expensive commission fees also discourage companies from using these services due to increased operating costs. Blockchain and smart contracts have brought promising hints to address the challenges in the data auditing and certification process. It can replace the current process of expensive and not fully trusted third-party data audit process, saving time and costs [REF-138].

2.2.7.1. Open challenges and needs

The advent of blockchain technology has opened new opportunities for secure, decentralized service certification and auditing, yet there remain several challenges that the REWIRE project must address to optimally leverage its potential. Blockchain's inherent immutability and decentralization offer robust solutions for data integrity and traceability, but limitations in pulling or pushing data from external systems

present a significant hurdle. The utilization of oracles, particularly secure decentralized ones combined with trusted hardware, has shown promise in addressing this issue. However, there are ongoing issues of scalability, efficiency, and privacy, which demand further research and development within the REWIRE framework. Moreover, striking a balance between on-chain and off-chain data management continues to pose challenges, particularly when handling large volumes of data in IoT applications. The use of smart contracts for automated and secure data collection could offer a solution yet requires careful integration with secure oracles. Additionally, while blockchain wallets and verifiable credentials provide robust mechanisms for access control and identity verification, ensuring these tools' compatibility and efficiency within the larger IoT ecosystem is another crucial task. Finally, the application of blockchain in certification and auditing is promising, yet the full potential of this technology in replacing conventional third-party services and reducing operational costs requires more investigation. The REWIRE project's focus on providing a comprehensive security assessment framework for IoT devices necessitates addressing these challenges to ensure secure and efficient data management, access control, and service certification within the IoT landscape.

2.2.8. Secure distributed service operation through misbehaviour detection

Presently many organisations rely on distributed service environments (e.g., IoT, Cloud computing, etc.) to provide critical services to their users; thus, it is of great importance to ensure their secure operation which in turn will allow such organisations to provide services to their users with greater confidence and peace of mind. Typically, distributed service environments are composed of multiple heterogeneous devices, distributed across different machines or nodes on a network that communicate with each other to provide various services to users. While this communication offers many benefits (e.g., scalability, flexibility, etc.), it also generates security and privacy challenges, as its difficult to ensure the security and correct behaviour of every device and service in the network, considering also that lately attacks have become more aggressive, and attackers have adopted a more strategic approach [REF-139]. Traditional security mechanisms rely on excluding external attackers who lack key credentials. However, if the malicious actor pretends to be a network node, these approaches are not very effective [REF-140] . Misbehaviour is a term commonly used instead of intrusions or attacks when addressing threats that are carried out by the participating nodes in distributed networks. In addition to attackers and malevolent participants, malfunctioning nodes are also included in the definition. Any node that transmits incorrect data while the hardware and software are acting as expected is considered to be misbehaving. Misbehaviour detection can help with these issues by offering an automated mechanism to find possible security concerns and inappropriate behaviour and take the necessary steps to lessen their consequences. Researching the relevant literature, three major categories of misbehaviour detection systems are identified [REF-139]:

Node-centric. In this category, the system checks each node's behaviour to be in line with the protocol specifications. They can be further divided into (a) behavioural-based in which the detection in performed based on abnormal actions, and (b) trust-based in which each node has a trust-value, and it drops below a predefined threshold it is detected as a threat.

Data-centric. In data-centric misbehaviour detection systems, the focus shifts away from the protocols and the model of the distributed network into the data that is shared between the nodes. Two subcategories are observed: (a) plausibility-based, in which checks are performed on the correctness of the data and (b) consistency-based, in which the relationship between the information that is shared is validated in order to decide the trustworthiness of each message.

Hybrid. In hybrid systems, a combined approach is adopted in order to evaluate each node based on the data that is shared and the correctness is decided using a data-centric detector.

In all of these categories, both the statistics of each node/device and the data that is shared between them can be used to train Machine Learning (ML) models in order to detect misbehaviour. Another classification [REF-141] can be made based on the mode of the misbehaviour detection:

Local detection. The detection is performed independently to each node based on internal consistency checks. This approach does not rely on the responses of other nodes, but checks for plausibility,

consistency and behaviour.

Collaborative detection. These systems use data detection schemes to monitor each node based on using the responses of neighbour nodes for misinformation.

Global detection. In this category, a back-end system is used, and the detection is not only based on local or collaborative schemes. This operation may involve gathering information from each device over predefined period.

AI-based misbehaviour detection in distributed IoT environments. In the next decade, it is projected that the Internet will be a seamless construction of common networks and connected devices [REF-142]. As a result, security becomes a hot problem in the IoT. Systemic security mechanisms and cryptographic security mechanisms are examples of existing security techniques. The possibility of network attacks, such as the volume of IoT service requests received in a short period of time or the availability of illegal access to specific services, may result in catastrophic failures [REF-143]. Therefore, in order to detect the threats, the use of misbehaviour detection is necessary for keeping the IoT networks secure and accessible. However, because of the sources and energy limitations of the IoT devices, it is often difficult to operate the complex misbehaviour detection techniques that are traditionally used. Therefore, together with the growing interest in Artificial Intelligence (AI), ML and its derivatives (e.g., deep learning) are becoming the focus of most research conducted on misbehaviour detection in distributed IoT systems. ML-based solutions can be divided into three subcategories [REF-139]:

Traditional learning. Refers to techniques that aren't based on deep learning. They can be further divided into:

1. **supervised learning, where** a labelled dataset is used to train an algorithm into classifying data or predicting the outcome. Supervised learning algorithms are classified into:
    a. **classification**, where the algorithm is used to classify data into specific categories, e.g., malicious or normal nodes. Commonly used models are Logistic Regression (LR), Support Vector Machine (SVM), Random Forest (RF), XGBoost, K-Nearest Neighbour (KNN), etc.
    b. **regression**, where the algorithm is used for example to predict the trust-value of a node based on a reputation system. Common models are Linear Regression (LR), Decision Tree (DT), etc.
2. **unsupervised learning,** where no labelled dataset exists and instead the trained model tries to find patterns in the data structure. Can be further divided into:
    a. **clustering**, used to divide data points that are similar into groups that are easier to understand (k-means, hierarchical clustering, etc.)
    b. **anomaly detection**, which identifies unexpected events or unusual items in a dataset without prior knowledge (One class SVM, Elliptic Envelope, Isolation Forest, etc.)
    c. **dimensionality reduction**, transforms data from a higher dimensional to a lower dimensional space without losing important information (PCA, etc.)

Deep Learning. A subset of ML that is based on deep artificial Neural Networks (NN), which have enabled advances in several applications due to their success especially in very large datasets with a high number of features. Can be further divided into

1. **supervised learning**, which include Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN) as the most common model architectures,
2. **unsupervised learning**, which includes the Autoencoder, a common architecture for anomaly detection in unlabelled datasets, that operates by compressing the input and reconstructing it. Calculating the reconstruction loss can be used to detect anomalies.

Advanced ML concepts.

1. **Federated Learning (FL),** a distributed training technique that is based on the collaboration of multiple nodes in order to build a global model without sharing the data outside of the nodes.
2. **Reinforcement Learning** is a goal-oriented learning algorithm that trains to maximize a reward over time.

3. **Generative Adversarial Networks (GAN**), which are used to create new data similar to the input and consist of two networks that are trained together, the generator and the discriminator.
4. **Transfer Learning** exploits the knowledge gained by solving a given problem to apply it in another related problem. For example, the knowledge of detecting DDoS attacks can be used to detect DDoS attacks in another network.
5. **Semi-supervised learning**, where the training process combines learning on a small amount of labelled data with a large amount of unlabelled data during training.

AI-based misbehaviour detection over system traces and attestation results. AI-based analysis of system traces and attestation results refers to the application of AI algorithms to these data sets in order to identify potential security issues or abnormalities. System traces are a log of system events (such as network traffic, file changes, log entries, etc.) that can give information about how a system behaves, whereas attestation results are a collection of measurements that confirm the security and integrity of a system's hardware, software, and firmware components. Network logs are thought of as an incredibly rich source of information in this area that may be used for a variety of tasks, including estimating the network's current operational health and identifying and preventing possibly hostile activity [REF-145]. To this end, ML models have been frequently used for network intrusion detection systems [REF-146] and can be divided into shallow ML and Deep Learning (DL) based on their model architecture.

These techniques can be further discriminated into a) supervised, b) semi-supervised and c) unsupervised learning based on the existence of anomalous labels on the training dataset. However real system traces and attestation results usually lack labelled datasets. On that matter, unsupervised anomaly detection systems can discriminate abnormal behaviour without the use of labelled data. In these situations, clustering-based strategies [REF-147][REF-148] have proven to be a practical solution; however, more advanced techniques, relying on Deep Neural Networks (DNNs), are being studied [REF-149] due to the vast and extremely heterogeneous structure of network traffic data. Eventually, the unsupervised network anomaly detection problem has been successfully addressed by generative models.

In a publication [REF-150], the authors used a newly published network traffic dataset, to develop a novel deep learning approach for misbehaviour detection using generic features at packet level. They evaluated their proposed schemes using accuracy, precision, recall and F1 measures and demonstrate significant results. Nevertheless, they highlight the lack of reliable datasets that include misbehaviour.

2.2.8.1. Open challenges and needs

All ML-based systems, have their challenges and risks that require the appropriate attention in order to ensure a smooth and robust operation, especially for critical tasks like misbehaviour detection.

Adversarial attacks: These systems are vulnerable to adversarial attacks, in which the attacker can infer vital information about the model in order to launch attacks that may poison its predictions [REF-144].

Quality of data: The models also need a sizable and trustworthy training set to work properly thus sometimes protection is not guaranteed against zero-day vulnerabilities. Also, data generated from various sensors and devices is usually diverse and needs a considerable effort in preprocessing and homogenisation.

Labelled data: Datasets annotated with misbehaviour examples are not often available in order to train a supervised classification model. Therefore, unsupervised learning techniques are utilised, although discriminating anomalous patterns from normal is not an easy task [REF-145].

Scalability: These ML-based solutions are relatively difficult to deploy and certify for embedded use. Also, scaling a real-time misbehaviour detection system to a network of thousands of devices can have serious effects on the performance and accuracy [REF-151].

Concept Drift: Ultimately, the model may need to be retrained in order to recognise novel, previously unidentified misbehaviour types due to environmental changes that tend to introduce a shift in the data distribution over time.

Explainability: In order for the operators to understand and take actions, the misbehaviour detector's predictions need to be interpretable and explainable.

Security: Access to sensitive data is often required for training the ML algorithms, leading to privacy concerns. Appropriate technologies need to be used in order to ensure the protection of this information.

Resources: In most cases, depending on the ML model architecture, these systems require resources that are not abundant in IoT devices, hence developing lightweight solutions while also maintaining the performance is very important.

Although these challenges are crucial, they do not affect the rapid adoption and development of ML-based techniques being used in misbehaviour detection systems. Nevertheless, addressing these challenges in safety critical and distributed environments such as smart cities or satellites is the key to create more dependable and adaptable systems with efficient monitoring and early warning.

2.3 Consortium's Shared Vision for REWIRE

Our vision for REWIRE is to enhance the security posture of next-generation smart connectivity "Systems-of-Systems" with the endmost goal of enabling a holistic security management framework that can safeguard IoT environments during the entire lifecycle, i.e., from the Design to the Runtime phases, capitalizing on trust-aware defence mechanisms that exploit emerging technologies based on Formal verification, Theorem Proofs, Open Standard Instruction Set Architectures (ISA), Trusted Computing, Blockchain and Artificial Intelligence (AI). The goal is to build on the combination of these technologies, as enablers for the secure and formally verified design and configuration, secure operation, trust orchestration, and verifiable computing of safety critical IoT components.

By coupling the zero-trust and security-by-design principles under the concept of "Never Trust, Always Verify", REWIRE will ensure the security and trust of embedded and IoT devices. In this defensive strategy, we envision four main phases leveraging these aforementioned key technologies: (i) Formal verification of SW & HW co-design and cryptographic protocols (ii) FW & SW security updates and patching validation (iii) Runtime attestation for verification of IoT devices' operational assurance using customizable lightweight TEEs, and (iv) Blockchain-assisted AI-based misbehaviour detection in a distributed fashion. In addition, REWIRE contributes to "Open-Source SW- and HW-Functional Description for Establishing Trust Anchors and Cyber-Security Governance in Systems-of-Systems" by exploiting metadata generated from application behavioural patterns and operations to determine appropriate security policies and properties to block potentially harmful instructions.

REWIRE aims to provide formal verification in the unexplored areas of HW and SW co-design and cryptographic systems. Thus, the vision of REWIRE towards providing flexible HW architectures and converging security and performance goes beyond the state-of-the-art methods that focus only on HW verification. REWIRE, also apart from formally verifying HW and SW co-design, envisions formally verifying the adopted crypto-schemes and their actual implementation, compared to other solutions that verify only the scheme and not the implementation. Towards this direction, REWIRE's goal is to provide a formally verified AES scheme (i.e., mode of operation) and implementation that is resistant to passive side-channel attacks and key leakage.

REWIRE should also support security by design and trust governance of resource-constrained embedded and IoT devices using open standards. Thus, the emerging RISC-V and keystone technologies are considered a good fit as the common underlying HW (i.e., RISIC-V) to support and extend TEE (i.e., keystone) functionalities. In parallel, we envision these technologies to be common to all the REWIRE use cases and also formally verified by the REWIRE mechanisms.

REWIRE will investigate the use of runtime monitoring of trustworthiness and operational assurance of SoS. To do so, the design of new bread of remote attestation schemes (e.g., CIV with static properties

and key restriction usage policies, CFA, etc.) will be supported by extending the underlying TEEs functionalities. Non-intrusive tracing techniques will be used in order to extract measurements from the device (e.g., sequence of assembly commands, etc.). The tracing and the attestation mechanisms will be more efficient since they focus on assembly commands that stem from ISA - a reduced set of instructions - of the RISC-V. In tandem, the combination of the co-design formal verification and the remote attestation will provide more efficient attestation results, since only specific properties will need to be attested. Further, REWIRE envisions supporting automated FW and SW validation based on vulnerability analysis and binary instrumentation.

Our aim is that REWIRE will provide a novel, practical, and scalable mechanism to support zero-touch onboarding aligned with the SSI concept toward creating a chain of trust. Thus, verifiable credentials containing the evidence of attestation results will be utilised to support DIDs. In parallel, blockchain technology will support ABAC and secure data sharing. Secure oracles will also ensure the data veracity of the stored information. Last but not least, REWIRE will provide misbehaviour detection and early warning in such distributed environments as the cornerstone of keeping secure and accessible safety-critical infrastructures such as smart cities. Our vision is that REWIRE will positively affect the life of everyday people by enabling them to achieve greater levels of security and trust of their always-connected activities.

Chapter 3
Methodology

This chapter is focused on the followed methodology for designing the REWRIE MVP, starting with the requirements definition process, to their elicitation methodology and finally the extraction of technical and use case requirements.

3.1 Methodology for MVP design

The term Minimum Viable Product (MVP) is a version of the product that includes enough features to be usable by end users who can then provide feedback for future product development. In other words, it is a product with enough features to attract early-adopters to validate the product idea early in the product development cycle. The term MVP was introduced by Frank Robinson back in 2001 and then popularized by Steve Blank and Eric Ries. Using an MVP to test a business model is probably the most popular technique.

REWIRE's MVP needs are different compared to a company's needs, since it is a Research and Innovation Actions (RIA) project aiming at reaching a Technology Readiness Level (TRL) equal to or higher than five. However, REWIRE's consortium common vision, has already set the goals of the project and agreed on the core features of the REWIRE framework even from the proposal phase. MVP should be the simplest version of REWIRE framework and deliver the core value proposition to end users and solve their main problem. This common vision includes a gradual MVP approach for the smooth implementation of the framework and its adoption from the end users. Also, the contribution of the use case partners to the design of the REWIRE's MVP towards a clear definition of the REWIRE's scope and purpose is of paramount importance. A collaborative process among all the consortium was followed, from the insights of the use case partners to the analysis of the technical and academic partners towards answering core questions and describing the "as-is" and "to-be" scenarios for each use case. This process will assist in identifying existing gaps and potential refinements in the project's vision.

3.2 Requirements Definition Process

REWIRE's consortium worked together to refine the value propositions of the project and agreed on the main services and underlying technologies (see Section 2.2), after conducting State-of-the-Art (SotA) analysis for the main research pillars and innovations of the project (see Section 2.3). This SotA analysis is of paramount importance due to the research nature of the project, while it allowed the consortium to better understand the current needs and state of the market. All this aggregated information regarding the market analysis will be utilized in the context of WP7, which is focused on the dissemination, standardization, exploitation, and impact creation of the project in order to promote REWIRE's outcomes and define a go-to-market strategy for both the REWIRE framework as a whole and its components individually.

In parallel, the consortium identified the stakeholders, the specific actors involved in the envisioned use cases, and the corresponding goals, in order to gain a better understanding of the intended beneficiaries that the REWIRE framework be of interest. On top of that, it is necessary to collect and prioritize both functional and non-functional requirements. In the context of REWIRE, the requirements are gathered in two ways. The first part of the requirements gathered from the technical partners who are responsible for designing and implementing the REWIRE framework and resulted in the technical requirements. The second part of the requirements was collected from the use case partners who are responsible for the demonstrators and resulted in the use case requirements. After the finalization of the collection process, the responsible partner aggregated and prioritized the requirements with the consensus of the whole consortium.

An MVP is not just a product, but a process of getting through the Build-Test-Learn (BTL) feedback loop with the minimum amount of product iterations and experiments. The BTL cycle is integrated into the development of the core technical WPs (e.g., WP3, WP4 and WP5) that involves the creation of both initial and final version of the project modules and components as well as the REWIRE framework itself. BTL approach maximizes the value added to each module/component per se and to the entire REWIRE framework. In a nutshell, MVP is in line with the overall consortium vision and the adopted development process. The main objective of the MVP is to provide tangible value, validate methodological ideas and concepts, and serve as a baseline for design and implementation activities. The REWIRE MVP is of paramount importance towards directing and shaping the ongoing work through the project's lifecycle. REWIRE's MVP methodology is depicted in Figure 3.1.

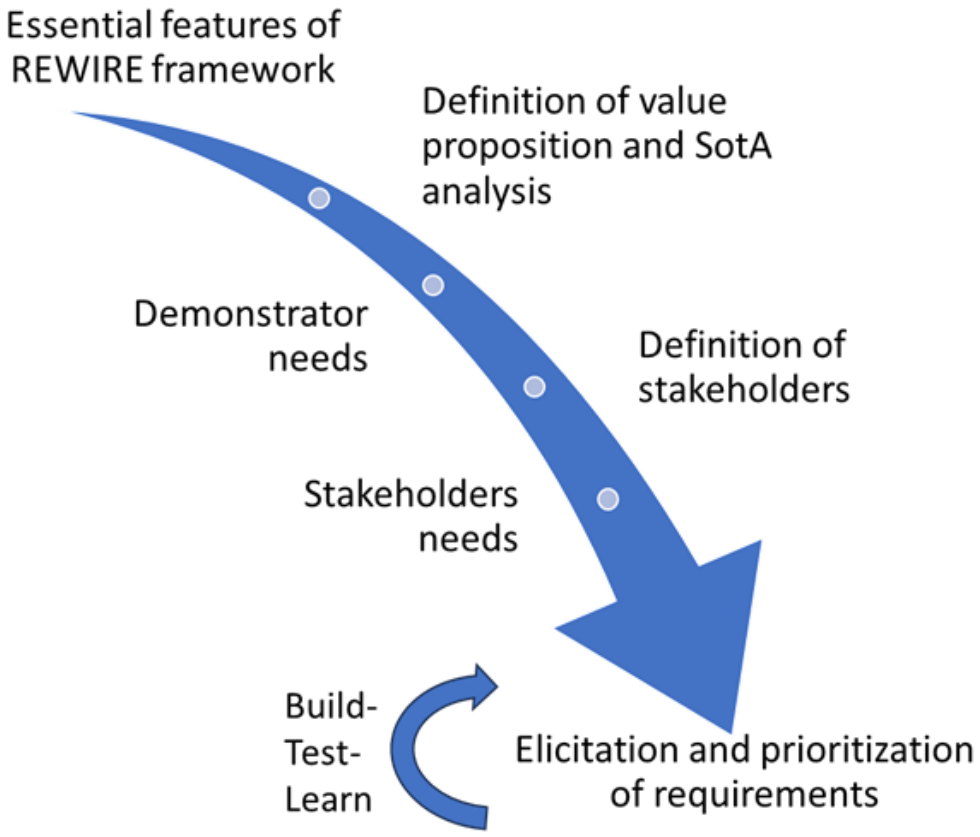

Figure 3.1: Methodology for definition of REWIRE MVP

3.3 Requirements Elicitation Methodology

The process of collecting requirements forms a substantial foundation for creating the REWIRE framework's business value. Requirements are a crucial component of the proposed REWIRE solution. Since these requirements form the basis for addressing identified needs, it is vital that they are specific, measurable, attainable, reasonable, and traceable. To collect them it is essential to actively involve all the relevant stakeholders and actively engage them in the process. Also, the elicitation of requirements is typically not a one-time process since the identification of gaps in the collected requirements may necessitate the elicitation of additional ones. The following subsections provide an overview of the employed methods within the REWIRE project for defining stakeholder requirements (e.g., technical and use case requirements).

3.4 Extracting Technical Requirements

The Agile framework is an iterative methodology and it's cantered around adaptive planning, self-organization, and short delivery times. Such a methodology is flexible, fast, and aims for continuous improvements in quality. Thus, why has REWIRE employed such a methodology. Agile emphasizes clear communication and understanding between the business, technical, and scientific aspects of the project, while establishing transparent expectations at the project's commencement (e.g., REWIRE's vision) and at each milestone, fostering collaboration and progress [REF-152].

In the context of REWIRE, system raw requirements were gathered with individual interviews conducted among the consortium technical partners. Raw requirements are the high-level requirements that have not undergone further analysis or been formally documented in a well-structured requirement notation [REF-152]. More specifically, REWIRE consortium run an iterative internal process for the requirement aggregation, using online tools for the interviews. This iterative process included brainstorming techniques, ad-hoc calls, and collaboration among the whole consortia. On top of that, the process results were mapped to the technical aspects of the framework and linked to the identified value propositions of REWIRE, leading to the identification and formulation of the necessary technical requirements. These are documented in Section 5. However, it is crucial to acknowledge that sometimes the requirements derived from the interview process were confusing and vague. This vagueness is caused by either stakeholder lacking technical expertise to understand and to provide accurate answers or by misinterpretation of the stakeholder needs by the system architects. Thus, the academic partners of REWIRE conducted a thorough analysis, starting from the state-of-the-art and literature review to the industry best practices. This analysis validated the requirements, identified potential applicable standards, and enhanced the quality of the collected raw requirements to bring value and meaning.

The experience and expertise of stakeholders is of paramount importance for the elicitation of requirements. However, it is crucial to acknowledge that sometimes the requirements derived from the interview process were confusing and vague. This vagueness is caused by either stakeholder lacking technical expertise to understand and to provide accurate answers or by misinterpretation of the stakeholder needs by the system architects. Thus, the academic partners of REWIRE conducted a thorough analysis, starting from the state-of-the-art and literature review to the industry best practices. This analysis validated the requirements, identified potential applicable standards, and enhanced the quality of the collected raw requirements to bring value and meaning. The monitoring of the requirement collection and extraction process was the main focus of the WP2. The partners involved conducted several telcos both bilateral and as whole to discuss, provide their feedback and evaluate the collected requirements.

3.5 Extracting Use Case Requirements

As already mentioned, apart from the technical requirements gathered from the stakeholders, the use case partners provided a detailed description of the use case requirements (e.g., user stores along with the technical explanation) and the core functionalities they intend to use of the REWIRE framework. Their feedback was of paramount importance for more accurate requirements that are tailored to the project's use cases also providing opportunities for further research. In this phase the use case partners collaborated closely with the research partners in order to refine the requirements and elaborate on the technical details derived from the narratives. More precisely, the refinement process included translation of the narratives into user stories, a high-level description of requirements. User stories provide concise units of ideas from the end-user perspective. Thus, there are also understandable from non-technical partners. Typically, the user stories are short (e.g., a single sentence) and self-explanatory with sufficient information for the requirement description.

Initially, the REWIRE use case partners provided user stories that describe the "to-be" reference scenario in order to extract, later in time, the requirements from these user stories. A user story, a) emphasizes the perspective of a specific role that will use or be affected by the story, b) defines the requirement in a way

that is meaningful to the role, c) clarifies the reasoning behind it, d) facilitates the definition of high-level requirements without delving into details that are confusing at this early stage and e) considers the end-user's goals and added value. Several user story textual templates exist online, however, in the context of REWIRE the Connextra template is adopted [REF-154]. An example of the textual template is the following: *As a* < ***type of user*** >*, I want to* < ***some goal*** > *so that* < ***some reason*** >

However, due to the nature of the agile project new or updated user stories may arise at any stage of development, ensuring a continual focus on important and meaningful to the end-users' aspects, while potentially excluding features that may have less importance in terms of the added value. In REWIRE towards assuring that the user stories meeting is of high quality, a validation process is incorporated in the methodology in order the user story to be aligned with the INVEST characteristics [REF-154]. INVEST acronym stands for Independent, Negotiable, Valuable, Estimable, Small, and Testable to help us remember guidelines for writing effective user stories. This methodology is used to quickly review the quality of the user story presented to the team. If a user story fails to satisfy one of these criteria, the team may consider rephrasing it or rewriting it. The user stories described for each of the three REWIRE use cases are presented in Section 6 as part of the corresponding "to-be" reference scenarios.

Chapter 4
REWIRE Conceptual Architecture and Functional Components

This chapter is dedicated to the REWIRE conceptual architecture and its functional components. In a nutshell, the following sub-sections provide the conceptual architecture, the two phases of the architecture, i.e., the Design-time and the Runtime phases. In addition, this chapter focuses also on the REWIRE-enabled edge devices and services workflows in order to shed the light on how the REWIRE framework provides the holistic security management framework for safeguarding the edge devices.

4.1 REWIRE Conceptual Architecture

The REWIRE conceptual architecture is illustrated in Figure 4.1. The architecture is divided into two major parts. On the left-hand side the design-time phase is illustrated, following a top to bottom approach on the description of the workflows that take place in that part of the architecture. On the right-hand side the runtime phase of the architecture presented, including the REWIRE-enabled edge device and the cloud-based backend infrastructure of the framework.

The following subsection elaborates on the high-level actions that take place in the context of the two major phases of the architecture. Sections 4.3 and 4.4 elaborate on the components that belong to the design-time and runtime phases respectively, while Section 4.5 focuses on the REWIRE-enabled edge device.

4.2 High-level Sequence of actions

4.2.1. REWIRE Design-time Phase

The design time refers to formal verification of the SW/HW co-design and the initial instantiation of the framework considering the initial phase of a deployment. In this phase, the security administrator, the use case providers and the OEM express the overarching requirements that need to hold on the system and are given as input to the formal verification processes of the SW/HW co-design, in order to analyse whether the system design can meet a desired level of security and trust by-design. Given the outcome of the formal verification, the security administrator will be in position to define additional security controls, as part of a Security Policy Management process, in order to support the formally verified system with additional security measures in order to further minimise the attack surface of the designed system. The design-time phase is illustrated at the left-hand side of Figure 4.1 and can be divided logically in the following steps.

4.2.1.1. Step 0: Definition of Requirements

The Security administrator defines the set of requirements which usually fall under different categories. In REWIRE, and until the moment of writing this deliverable, the technical members have identified three major categories of requirements, referred to them as the "***Overarching Requirements"*** of the system design that need to be considered. The overarching requirements include, but are not limited to, the requirements for the *Security Models*, the *Cryptosystems* and the *Binary Instrumentation*. The Security Models refer to any security-related mechanism that a system needs to integrate in order to meet its operational objectives in a secure and reliable manner. In the context of REWIRE, as Security models we define the mechanisms of the Cryptographic Key Management, the Zero-trust Onboarding, the Attestation Schemes, the Device State Management (function migration/isolation), etc. Each of the aforementioned

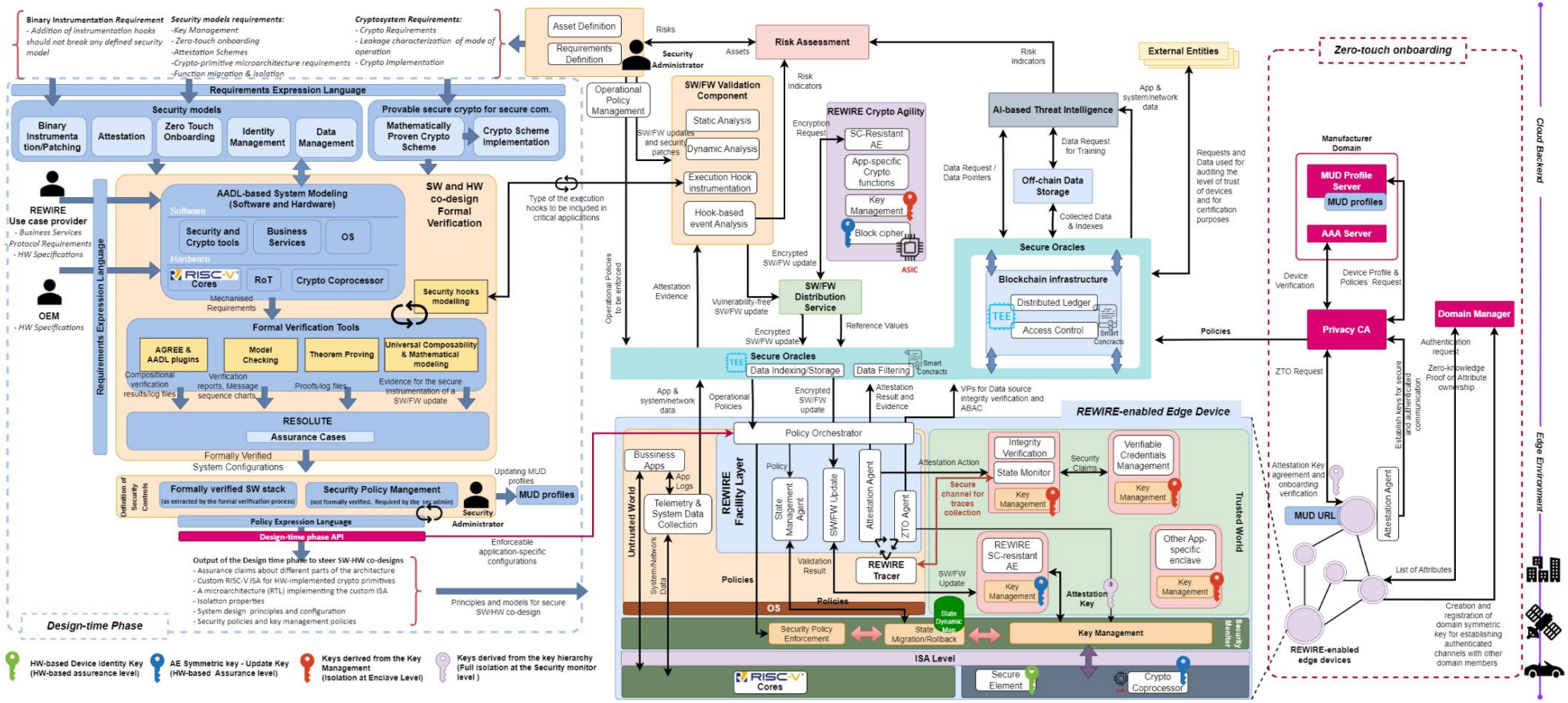

Figure 4.1: REWIRE Conceptual Architecture

mechanisms may entail to a number of mechanism-specific requirements that the system needs to take into consideration during its design phase. The Cryptosystem requirements refer to those that instruct the way that the crypto schemes will be designed and be included in the system design. For REWIRE, we particularly focus on side-channel resistant modes of operation in order to safeguard the SW update process from side-channel attacks. That is why in the architecture of Figure 4.1 the *Provable secure crypto for secure communication* is highlighted as a distinct component and it is not presented as part of the rest of the security models of REWIRE. Last, but not least, the Binary Instrumentation requirements are used in order to express the need for the addition of SW instrumentation hooks which will not break the secure and trustful design of the system. It has to be noted, that these categories are just an indicative subset or requirements which are applicable to the REWIRE project. By generalising this concept, the REWIRE design-time phase suggest that any requirement that needs to be met by a system design can be given as input during this phase so that to be considered by the formal verification process. A requirements expression language will be used and will be defined later in the project in order to be able to express the set of requirements and be given as input to the AADL-based System Modelling in order to be mechanised and feed the formal verification processes.

4.2.1.2. Step 1: AADL-based System Modelling

System modelling is a proprietary step before the actual formal verification process. Thus, a model-based design framework for Validation and Verification of a system's architecture is needed so that to model how the requirements map to specific system artefacts and verifying functional and requirements-driven correctness.

To do so, REWIRE uses AADL-based System Modelling, for both software and hardware, as the Model-Based Engineering (MBE) approach that will be used for specifying both software and hardware configurations, and, thus, building a representation of the system to be validated. This is what Figure 4.1 aims to represent in the AADL-based System Modelling component. In this component an abstract illustration of the system is represented, comprised from the SW and HW elements that synthesise its SW/HW stack. AADL is a standardized architecture description language, able to model hierarchical structure and connections among software and hardware components. In this step of the design-time phase, AADL is intended to be used in order to model a repository of proof artifacts for the various (security) properties related to the project use cases. AADL is used to define component properties. Requirements can be also defined as properties which can then be associated with the various system components in order to map requirements to the system architecture.
This mechanised representation of the system and the mapping of its internal components and requirements are given as input in the formal verification process (and the respective tools) in order to check the satisfiability of security requirements.

4.2.1.3. Step 2: Formal Verification

Once the AADL model has been generated, the REWIRE design-time phase proceeds to the actual formal verification process. REWIRE will use and combine several tools and methods in order to perform the formal verification. At the moment of writing this deliverable, Model checking, Theorem Proving and Universal composability have been chosen as the approaches that can cover the spectrum of formal verification activities of REWIRE. Each of the aforementioned methods focuses on a specific aspect of the formal verification. For instance, Universal Composability will be used for the validation of cryptographic protocols used in REWIRE for authentication and authorisation, Theorem proving will be used for the validation of the implementation correctness of the side-channel resistant mode of operation of the REWIRE authenticated encryption scheme, while Model Checking will be used for validating the correctness of the REWIRE attestation schemes and the monitoring hooks. It has to be stated that this approach may change as the project progresses. The final association between the methods and the artifacts to be verified will be documented in D3.1, as the aim of this section is just to elaborate on the architectural aspects of the formal verification layer of the design-time phase.

Special mention is needed to highlight also the modelling of the monitoring hooks. Security hooks will be used in REWIRE for instrumenting critical security processed of a FW/SW in order to be able to monitor

in an efficient manner its execution during runtime (See Section 4.4.6). However, the instrumentation of a binary is not a straightforward process, while the addition of monitoring hooks may even break the security model of a system. Thus, REWIRE will carefully treat the process of adding monitoring hooks, by considering this process as part of the overall formal verification process. The security hooks modelling is actually an iterative process which is being executed until the inclusion of the necessary hook is included in the FW/SW and the formal verification process results to a security model that holds the necessary security and trust properties. At the moment of writing this deliverable, finite-state machines is considered the most prominent technique that can be used for the modelling of the monitoring hooks.

The above methods generated as outputs a set of compositional verification results, Proofs on the satisfiability of security and trust requirements and evidence on the correctness of the attested schemes. Resolute is used to collect information about the requirement satisfaction. If all the required artifacts are present, the tool outputs that the top-level system is secure, confirming that the security goals are supported by evidence. Otherwise, it pinpoints to the particular component from which the relevant proof artefacts are missing and need to be provided.

4.2.1.4. Step 3: Definition of security control and Security policy management and enforcement

The REWIRE architecture will focus on mechanizing the formal verification outputs into enforceable security policies that will govern the device operation throughout the executed use-case scenarios. This will be achieved as part of the layer for the definition of security controls. In this layer, a semi-automated activity is followed. The security administrator interprets the outcome of the formal verification process so that to identify the part of the system which has not been validated, i.e., the part for which no security and/or trust guarantees exist for the security and trustworthiness of the system as a outcome of the formal verification. This means that for the rest of the system, for which it not possible to derive a formal verification of its correctness or trustful operation, the security administrator needs to define a set of mitigation actions or security policies which can safeguard the operation of the system during runtime. REWIRE will be focusing on the deployment of attestation policies that could be used in order to detect potential compromised components, binaries or deviations from the legitimate behaviour profile of critical applications.

Following this approach, the design-time phase of REWIRE, will result to a provably secure-by-design SW/HW co-design, covering the most security- and safety-critical operations of the system, as instructed by the defined overarching requirements, while the rest of the system will be protected through the definition of security policies defined by the system administrator.

The policies which are defined at the end of this process will take the form of enforceable policies that can be loaded on the devices and will regulate its operational behaviour during runtime. For this step, REWIRE will investigate policy expression languages which can support the project's objective, while an API will be designed in order to enable the actual enforcement of policies (when possible) on the devices through the policy orchestrator component.

Of course, the outcome of this process of the design-time phase is not only the dynamic enforcement of policies, but its is the final step of the formal verification actions for delivering a secure-by-design system. This means that the output of the whole process will be the definition of the assurance claims about different parts of the architecture, the definition of the RISC-V ISA for HW-implemented crypto primitives, the isolation properties of critical system applications, the security and key management processes. All the aforementioned aspects should be taken into account for the system design.

4.2.1.5. Step 4: Definition of the MUD profiles

The last step of the design-time phase is the definition of the Manufacturer Usage Description, or MUD,

profile of the system. The MUD profile for each device is ostensibly a set of rules and expected behaviours against which we can compare its current behaviour and identify any deviation. MUD (RFC 8520) is a standard for enabling IoT devices to signal to the network what access and protection they need, now ratified by the Internet Engineering Task Force (IETF). The purpose of which – as the name suggests – is to provide facility and standardisation for Internet of Things (IoT) manufacturers to declare the various functionalities of their respective devices.

Within the MUD standard, manufacturers dictate explicitly the precise, and only, behaviours permissible for their devices as dictated by the data-types and methods that are necessary for the IoT device to function as intended. Hence, the outcomes of the REWIRE design-time phase, through the formal verification of the SW/HW codesign and the definition of the additional security policies on behalf of the administrator, can be used in order to define the set of rules that represent the expected operational behaviour of a device. During the runtime operation of the system, and more specifically, as part of the zero-trust onboarding process, the security administrator of an operational environment can acquire the MUD profile, as this has been augmented by the REWIRE design-time phase.

4.2.2. REWIRE Runtime Phase

During runtime, the REWIRE framework offers a wide spectrum of functionalities in order to support the life cycle of the IoT deployments, starting from the secure onboarding and bootstrapping to operation, update, and decommissioning. For all the aforementioned phase, REWIRE offers functionalities that capitalise both on services running at the cloud-based backend of the framework, as well as on agents running on the end-devices. Given this, the workflows and execution steps of the runtime phase are not sequential, as was the case for the design-time phase which consists of 5 distinct steps, they can be seen as standalone functionalities which, to a certain level, may be depended among each other. Hence, we identify bellow the various workflows of the runtime phase and we offer a high-level description of the individual steps that are executed in each workflow.

4.2.2.1. Zero-touch onboarding

The secure onboarding of devices into the REWIRE network will cover the trust-aware enrolment to ensure that only devices which are in a correct state can join the network. The ZTO architecture is highlighted in Figure 4.1 on the right-hand side. The ZTO is the first action that needs to be performed in order to enrol first the device into the network and then, it can participate to the rest of the services which are described below.

REWIRE devices are enacting the ZTO through the attestation agent component. The latter is aware of the MUD URL. The device makes a contact to the Privacy CA first, which is responsible for activating the keys that will be used on the device for the secure communication. Once the communication keys are activated, ZTO request takes place, and the Privacy CA actually, based on the MUD URL, sends the request to the AAA Server. The AAA Server, which is the entity that manages the identity services of the manufacturer (in the manufacturing domain), passes this URL to the MUD Profile Server, which controls the verification process and the validation of the MUD profile. The MUD Profile server, which is aware of the MUD profiles of the devices, as those have been created by the manufacturer during the design-time phase of REWIRE, verifies the MUD file by asserting that the MUD file was produced by the device manufacturer and that it corresponds to the device that sent it to the MUD Profile server.

Given the MUD File, the MUD Profile server will translate it to a set of context-specific policies which are passed back to the Privacy CA. The latter then enforces the policies to the REWIRE blockchain infrastructure for that to enable the trust-aware continuous authorisation and authentication processes of REWIRE and the attribute assess control mechanisms. Once this process is completed, the device has enabled all the necessary credentials and the REWIRE infrastructure is aware of the policies that portray the operational behaviour of the device. More specifically, the ZTO process concludes by creating the Attestation Key on the device that will allow to perform the necessary attestation actions to prove its correct state when onboarding to the application domain. In fact, this is the next step, where the device

communicates with the domain manager, it provides its credentials in order to acquire the symmetric key that will allow the device to communicate with the rest of the devices in the context of the REWIRE architecture. More details for the ZTO are given in Section 4.4.7.

4.2.2.2. Secure Firmware and Software validation and update

One of the core functionalities offered by REWIRE is the secure FW/FW update process. In today's IoT deployments there is an urgent need for scalable and secure FW/SW update processes for ensuring that the IoT deployments are always up to date, they operate in a legitimate way, and that the vulnerability patching may occur over-the-air so that to ensure the fast and at-scale deployment of patches and new FW/SW versions. Towards this goal, REWIRE creates a unique workflow consisting of various stand-alone technical artifacts which, when combined, form a vital service for guaranteeing the security and continuity of the IoT deployments.

More specifically, the SW/FW validation component, the SW/FW Distribution service, the Side-channel resistant mode of operation for authenticated encryption, and the Secure oracles operate in tandem for the provision of the Secure FW/SW validation and update service of REWIRE. As illustrated in Figure 4.1, the security administrator, decides to initiate the SW/FW update process. This process may be triggered by the administrator after analysing the security posture of the infrastructure, based on the events collected through the various mechanisms of the REWIRE and the risks identified by the risk assessment process. REWIRE offers a complete tool for validating the secure implementation of a FW/SW, analysing its source code to detect potential vulnerabilities through static and dynamic analysis methods (see Section 4.4.6). In addition, specifically for some limited number of safety-critical operations, the REWIRE SW/FW validation Component is able to instrument the FW/SW for adding monitoring execution hooks which will enable the efficient introspection of the operational profile of the process while being executed on the device. Given this, efficient attestation of the applications behavioural profile can be achieved during runtime.

After ensuring that a FW/SW is free of known vulnerabilities, the SW/FW distribution service undertakes it deployment. The latter is not a service offered by REWIRE, but we are going to take advantage of the existing updating services that the use case partners have already in their environments. REWIRE focuses on the secure SW/FW deployment. That is why, the SC-Resistant AE of REWIRE is used in order to encrypt the update and ensure it integrity and authenticity. Notably, these properties are achieved, while the specially designed mode of operation of the authenticated encryption scheme protects again side-channels, making sure that even the most sophisticated attacker that can physically monitor the process, cannot leak the symmetric key used for the encryption.

After the encryption, the FW/SW distribution takes place. Depending on the requirements of the operational environment, this can be performed in an "1-to-1" manner, suggesting that the SW/FW distribution service enforces the update directly to the end-device, or in an "1-to-many" approach, where the secure oracles undertake the distribution of the update to multiple devices which may be geographically distributed (e.g., in smart cities.) Regardless of the way that the update is distributed, the end-devices have the necessary cryptographic capabilities in order to decrypt the update, validate its integrity and authenticity and initiate securely the update instalment process, by taking advantage of the isolation mechanisms offered by the REWIRE TEE. In this way, REWIRE ensured the secure and FW/SW update process, from the beginning of process on behalf of the administrator, until the moment of the actual deployment.

4.2.2.3. AI-based threat intelligence for the detection of anomalous events

REWIRE offers an AI-based service for the detection of abnormal incidents based on monitoring data that reflect the operational behaviour of the end-devices. The service requires the synergy of several technical components of the architecture, as those are illustrated in Figure 4.1. More specifically, the AI-based threat intelligence component is the heart of this service. This is where the actual AI model is trained and instantiated. This component works in synergy with the off-chain-data storage for retrieving data for training purposes and the Secure Oracles and the blockchain infrastructure in order to acquire data during runtime. These data are being collected by special agents installed on the end-devices, collecting telemetry and system data that can reflect the security state of the device. It has to be noted, that the data collection agents are not defined in the context of REWIRE. We capitalise on existing solutions and on

tools that the use case partners of REWIRE have already in their environments.

As illustrated in Figure 4.1, the App & System/Network data captured on the devices is sent to the oracles, where a first step of data filtering and normalisation is applied before those are sent back to the Off-chain Data Storage to be saved and indexed, or, during the inference model of the AI model, as sent to the AI-based Threat Intelligence engine to be analysed. In case the engine detects any deviation from the legitimate behavioural profile of a device/system/critical function, a risk indicator is generated and is forwarded to the Risk Assessment engine in order to visualise the detected events, increase the awareness of the security administrator and identify potential risks. The detection of anomalous events may imply the need for the deployment of new security and operational policies or the need to the deployment of the FW/SW update/patch.

4.2.2.4. System introspection and operational assurance

One of the core offerings of REWIRE is a new breed of attestation mechanisms that will guarantee the operational assurance of the end-devices and their critical operations. In fact, the attestation mechanisms are a core defence measure to protect the IoT deployments in REWIRE, as it enables the detection of compromised devices and misbehaving processes. More specifically, as aforementioned in secure FW/SW validation service, one of the core enablers for the efficient attestation in REWIRE is the addition of the monitoring hooks. The latter enable the REWIRE tracer to efficiently monitor the execution behaviour of a critical application and collect the execution evidence in a targeted manner, as the tracer collects only those traces that the hooks have been designed to intercept. In this way, the extracted traces are far less, and the processing and decoding process requires less time. As highlighted in Figure 4.1, the REWIRE Tracer, the Attestation Agent and the ZTO agent operate synergistically. The attestation agent is responsible for initiating an attestation process and communicates with the integrity verification enclave to perform the system measurement in a secure and isolated manner in the trusted world. The enclave communicates with the Tracer in order to collect the traces from the untrusted world and passes this information back to the enclave (trusted world) in order to perform the final measurement and calculate the hash. REWIRE will use a local attestation protocol with key restriction usage policy. This means that only if the system is in a correct state, the underline RoT of the device will allow the system to use the cryptographic key in order to sign the measurement. More details on the attestation protocol are given in Section 4.5.5.

We need to highlight the role of attestation in the trust-aware continuous authentication and authorisation processes of REWIRE and in the ZTO process. As aforementioned, the authentication and the onboarding of devices requires evidence on the correct state of the device. That is, the attestation outcome is treated as a security claim which is transformed into a verifiable presentation using the Verifiable Credentials Management enclave in the TEE. The security claims, which prove whether a device is in a correct state or not, are used during the ZTO or any interaction with the REWIRE blockchain and are used as enablers for Data source integrity verification and as attributes for attribute-based access control. This REWIRE functionality implies that only trusted entities will take part in the REWIRE environment.

In parallel with the above-mentioned operations, the attestation evidence is collected by the attestation agent and is sent, through the Secure oracles, back to the REWIRE back-end infrastructure for further analysis and for keeping track of the security posture of the devices. More specifically, the attestation outcomes and traces are stored and indexed on the off-chain data storage so that to be used as evidence for certification and auditing purposes. In addition, the attestation evidence is sent back to the SW/FW validation component to perform a hook-based event analysis in order to analyse further the event and trigger an alert to the risk assessment component in order to increase the administrator's awareness for the detection of a compromised device.

4.2.2.5. Device state management

Another novel offering of REWIRE is a device state management service which aims to increase the resilience of critical services and contribute to the continuity of business services. More specifically, even

if we try to enhance the security of systems with innovative security solutions like the formal verification to achieve assurance by-design, and a wide range of techniques to safeguard systems during runtime (e.g., AI-based threat intelligence, Attestation, etc.), it is inevitable for devices to get compromised or to present malefactions due to system faults or misconfigurations. That is, REWIRE introduces a device state management approach, based on the use of TEE, in order to keep a critical service operational.
Two functionalities are offered to support this vision, namely the Enclave Migration and State fallback. The former suggests that whenever a critical service is detected as compromised (e.g. based on an attestation outcome) the state of the service (i.e. cryptographic keys and/or operational data) will be migrated to a new enclave in order to continue or re-initiate the service execution. The state migration between enclaves of the TEE is an innovative concept that will be documented in the context of the actions of WP4. More details can be found in Section 4.5.3. The migration process and its details (e.g. which process will be migrated from enclave A to B and under which circumstances) are instructed to the systems through operational policies which are being enforced on the device from the security administrator through the policy orchestrator of the REWIRE facility layer.
A similar approach is followed in the State Rollback case, where the system is reverted to a previous, but operational, state in case a system failure or compromise is detected. This operation is vital, especially for cases like the smart satellites, where physical intervention is not possible and a system failure could completely disrupt the service and/or the infrastructure. The state rollback functionality is regulated by the operational policies defined by the security administrator while the previous legitimate states of the system are stored on the State Dynamic Map component of the device. Both functionalities will be based on the newly designed feature on the REWIRE customisable TEE.

4.2.2.6. Risk Assessment and user awareness

This service will offer an assessment tool, which will be based on UBITECH's OLISTIC Cybersecurity & Privacy Risk Assessment engine and will focus both on the Design and Runtime phases of the deployments' lifecycle. Especially for the latter phase, it will provide near real-time evaluation of identified cybersecurity risks. The tool will be empowered with a qualitative and quantitative risk estimation method being in-line with the threat landscape of the IoT deployments and the REWIRE architecture. The method and tool will consider the dependencies that may exist in the IoT deployments to identify possible vulnerability paths that can result to propagated risks. The tool will be fed with information/evidence stemming from the monitored devices as a result of the security validations performed by the security mechanisms of REWIRE, i.e., Runtime Attestation and the AI-based misbehaviour detection. The end-goal is to increase cybersecurity awareness of the responsible operator so that to take informed decisions on updating and patching critical SW and FW resources, and identify risks from deviations of the behaviour profile of devices. The tool will provide a complete dashboard where the administrator will be in position to form a digital representation of the monitored environment and will be supported by visualisation tools for representing the identified risks.

4.2.2.7. Secure, transparent and accountable data sharing & access control

This service refers to the functionalities offered based on the use of the REWIRE blockchain infrastructure and the secure oracles. In fact, the BC infrastructure is the most critical component of the REWIRE runtime architecture as it facilitates the vast majority of the data sharing operations that have been described in the context of the other services/functionalities. As can be seen in Figure 4.1, all the surrounding components, either those belonging to the REWIRE framework or they constitute external entities, they are interfacing with the Secure Oracles in order to make transactions with the blockchain. Secure oracles undertake critical operations for enabling data sharing among the REWIRE framework, while in parallel are in position to apply data indexing, filtering, and preprocessing, depending on the type of data being collected. The logic of applying data processing and orchestrating the whole data sharing process is regulated through the use of smart contracts, while the oracles make use of a TEE in order to isolate critical data processing operations.

As illustrated in Figure 4.1, the blockchain infrastructure is used for the sharing and exchanging:

- **App & system/network data** to be fed to the AI-based threat intelligence component and to be

stored and indexed by the off-chain data storage.

- **Operational policies**, which are being pushed from the administrator and need to be enforced on the devices so that to regulate critical processes like the attestation, the device state management, etc.
- **Encrypted FW/SW updates** which are being forwarded to the devices, through the oracles, in the context of the SW/FW update process of REWIRE.
- **Attestation Results and Evidence** which are the outcomes of the introspection and attestation processes. These data are being forwarded to the Hook-based event analysis and to the off-chain data storage. These data can be shared also with external entities who may wish to access the data in order to check the security posture of the devices for certification purposes.
- **VPs for Data source integrity verification and ABAC.** As aforementioned, VPs are generated by the Verifiable Credentials management enclave as part of the ZTO and whenever and attestation process is performed. The VPs contain security claims which are used to exhibit that a device is in a secure state, and act as enablers for applying attribute-based access control in the contest of the trust-aware continuous authentication and authorisation services of REWIRE.
- **Any data,** as the blockchain infrastructure may be used for sharing and persisting any kind of data that the REWIRE framework or the IoT deployment might need.

We need to highlight that one of the main purposes of enabling the blockchain-based data sharing in REWIRE is the ability to perform data transactions in a secure, transparent and accountable manner. The secure oracles allow data transactions at scale, while the use of a TEE on the oracles can provide guarantees on the security, data provenance and data veracity. Last but not least, the use of the blockchain infrastructure enables decentralized data integrity and secure access control using attribute-based access control (ABAC), utilizing the VPs and the security claims as the enabling attributes.

4.3 Design Phase Architecture and Functional Components

While the previous section was focused on the description of the high-level sequence of actions of the design-time and runtime phase of REWIRE, the following sections focus on functional components per se, elaborating on the details of their design and their functional objectives.

4.3.1. Collection and Mechanisation of Requirements

4.3.1.1. Description and functional objective

Requirements definition, elicitation, and formalization are considered key to automate and apply Validation and Verification (V&V) approaches in order to prove that the developed system will meet its specifications. Formal methods practices can sometimes be challenging to adopt in industrial environments especially when multi-disciplinary engineers contribute to the end-requirements documents. On the other hand, the need for formalization and verification in the design of secure systems is now more evident than ever. To the end of easing integration of formal methods in REWIRE, we intend to use a model-based system engineering (MBSE) approach documenting the requirements using natural language, following though a series of patterns order to aid in the formalization process of the requirements. The latter will be used by the REWIRE V&V framework to verify the requirement against formal specification model (of SW or HW modules) for which certain security, safety, or correctness properties need to hold. The developed approach is an end-to-end solution, starting with natural language requirements as input and going all the way down to generated Temporal specifications (e.g., in model checking) or theorems (in theorem proving). Additional requirements may be defined as contracts within the OSATE framework attempting their verification using compositional verification. We employ an ontology-based reasoning approach using the RESOLUTE tool to link evidence of verification tools with the associated requirements using well-formed, sound assurance cases based on the SACM standard.

Such an approach will drastically reduce the required requirement traceability workload for both legacy and new requirements during the system development cycle.

4.3.1.2. Component workflows

In a typical workflow, as recommended by industrial standards, requirements are developed in a hierarchical fashion, based on the level of abstraction. First, intended functionalities of the system under account are described, e.g., in structured natural language (system level). Second, subsystems are identified within the overall system architecture, and software requirements are distinguished from hardware requirements (higher level). A third refinement step leads to the development of lower-level requirements, from which source code, on one side, and hardware components, on the other side, are obtained (implementation).

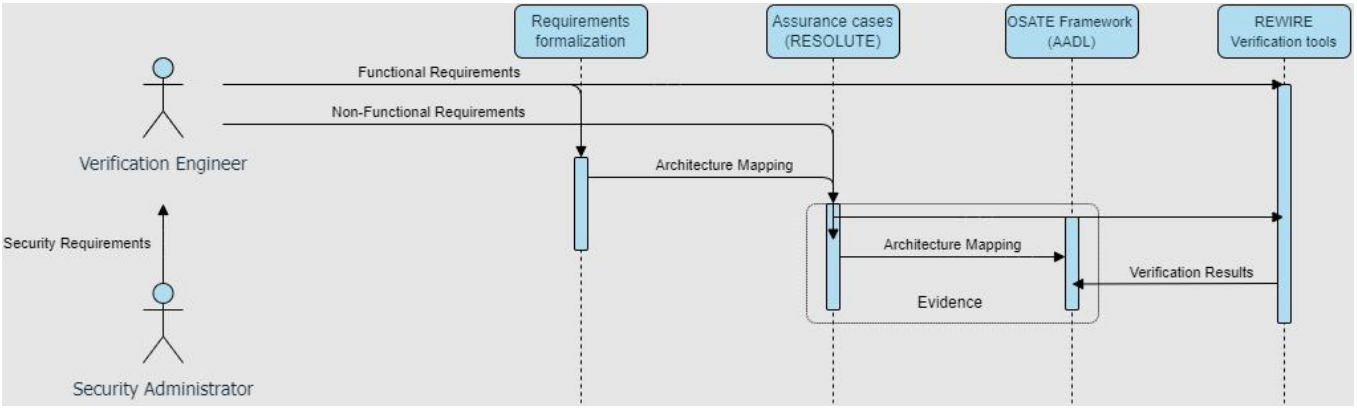

Figure 4.2: Collection and mechanisation of requirements sequence diagram

At the different levels, requirements can be expressed by means of different languages. The adoption of a formal notation, and possibly the use of executable models, according to the MBSE paradigm, provide numerous benefits for what concerns both validation and verification of requirements. Formalization removes ambiguity and helps determining whether a set of requirements is both necessary and sufficient to represent the intended functionalities; consistency, as well as correct derivation of a lower level of requirements from a higher level, can be (semi)automatically verified if the requirements are for example encoded in a fragment of first order logic, via tools such as model checkers and theorem provers. Complementary to formal verification, the use of executable models allows the realization of (possibly automated) testing strategies.

4.3.2. Provable secure crypto

4.3.2.1. Description and functional objective

The objective of REWIRE is to provide a provably secure cryptographic scheme in order to safeguard the critical operation of the secure SW/FW update. That is why, this component is highlighted as a stand-alone component in the design-time phase of REWIRE. The main requirement is to ensure that the update is transferred in a secure manner, even in the presence of side-channel attacks. More precisely, the SW/FW update transmission should provide authenticity (i.e., no attacker – even using side-channel attacks – should be able to modify the payload without being detected) and, in some use cases, confidentiality (i.e., no attacker should be able to read the payload's content, although, for efficiency purpose, side-channel attacks are not taken into account here). Also, the crypto solution should be easily deployable to a large class of IoT platforms and, therefore, should not depend on the availability of some carefully engineered component or some special design expertise. The side-channel security should be an artifact of the design and should depend on minimal and easily testable physical assumptions.

4.3.2.2. Component workflows

The security component to be developed in REWIRE is an Authenticated Encryption (AE) algorithm. It achieves security against passive side-channel attacks by construction, based on certain physical

assumptions. The main physical assumption in this case is access to a fully parallel hardware AES core (Crypto Coprocessor in the Figure 4.1), and an exclusive long-term secret key for the AES. The AE mode ensures that the core is accessed at most two times with a fixed key, and the physical assumption is that the hardware AES does not reveal the key in two measurements. The long-term secret should only be accessed by the hardware AES core in order to prevent any unwanted leakage. This key has to be pre-established and can be put inside the device during manufacturing at the factory level. Other than this AES core (with this long-term, pre-established secret), the rest of the operations in the AE can have unbounded leakage while still ensuring integrity.

As aforementioned, the AE scheme will be used to support the secure FW/SW update operations of REWIRE. Two different modes of operation of the update process will be supported:

1. One-to-one SW Update Case:

Authenticated Encryption by SW/FW distribution service:

The production of the secure SW/FW update is performed by the SW/FW update provider following the mode of operation specification. Depending on the physical security level of the environment in which the software update is encrypted, an AES coprocessor can be used (as depicted on the right of Figure 4.3) or a relaxation can be to use a software AES if side-channel attacks are not a threat in the SW/FW generation environment. The execution of the algorithm will yield to a pair (i.e., tag, ciphertext) that will only be valid if decrypted as described in the next section.

Key Management:

The SW/FW update key is assumed to be pre-established and not to be touched by the key management firmware or not to be transmitted online. To achieve this, a simple and secure enrolment protocol can be devised. After the device is manufactured, the key can be enrolled in a small non-volatile memory only read-accessible by the AES core through the AEAD firmware. The key enrolling that we denote as "*key personalization*" can be done with simple methods such as using a scratch card and putting the 16-byte string on it in the device. This must happen in a trusted environment. Below we point out the three main advantages of this approach:

a) The key is directly handled by the AEAD firmware which is already a part of a mathematically proven algorithm and formally checked implementation.
b) The enrolment phase happens only once in a trusted facility and a side-channel attack during this one-time operation (that too in a trusted environment) is hard.
c) The software provider is the only one to gain access to the SW/FW update key, he can set up both the server and the edge device in a manner that no trusted party is required.

Decryption on receiver's side:

As shown in Figure 4.3 below, the encrypted SW/FW update arrives through some channel and is forwarded to the AE engine. The AE performs several encryption (and possibly one decryption) queries to the AES core, among which only two queries are with the long-term secret. The rest of the queries are ephemeral secrets. In simple words, the AE first generates a session key and then encrypts/decrypts the message blocks by generating more ephemeral keys from this session key. If the verification of the update is unsuccessful, the AE engine returns a decryption failure message. Otherwise, it returns the decrypted update. Upon receiving the failure/success message the device sends an acknowledge (or failure) message back. This message is also encrypted and authenticated, with the same algorithm and key. The encryption process is exactly the same as that performed by the SW/FW distribution service, as described above. The encryption side is exactly same as the decryption side except that it outputs a ciphertext and a tag instead of success/failure messages.

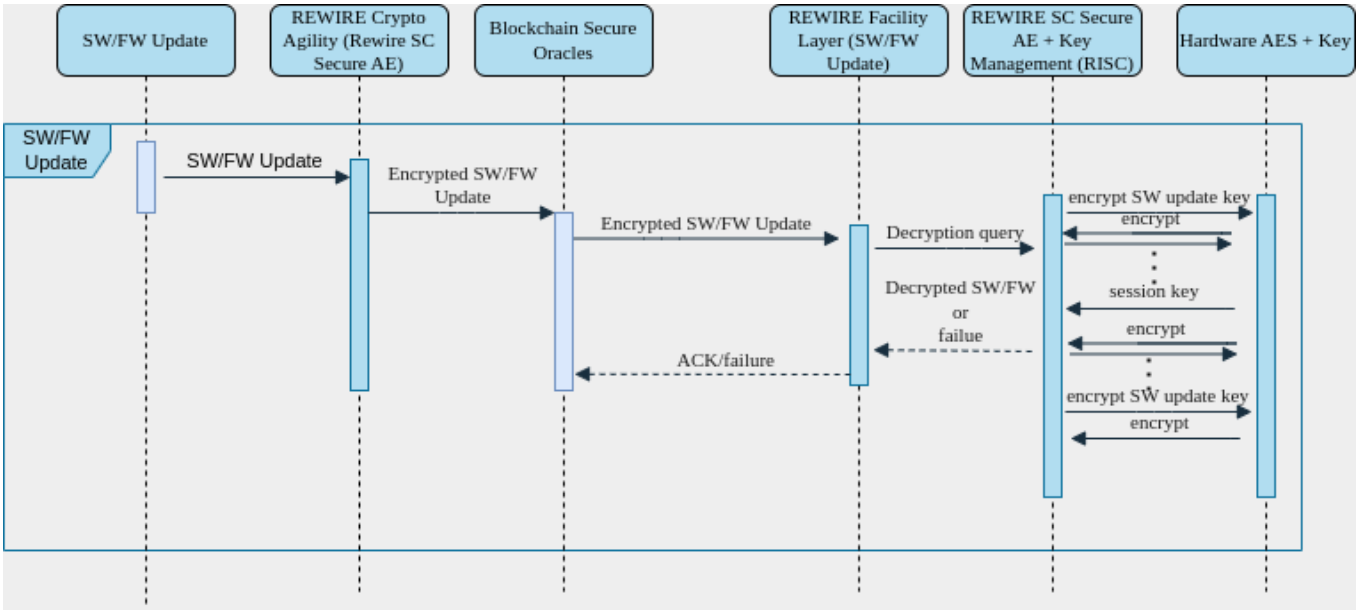

Figure 4.3: SW/FW update using AE (one-to-one)

2. One-to-many SW Update Use-Case:

This is a special use-case where the SW/FW is sent to many clients simultaneously, using asymmetric cryptography. Here the SW/FW distribution service possesses a public/private key pair. The update is signed by the private key. We assume that there exists a standard infrastructure to distribute the public key. The SW/FW distribution service signs the update with its private key, and all client devices can then verify the signature of the update using the corresponding public key. This operation does not require need any side-channel security to ensure integrity and authentication (as no secret is involved on the client side, and the server side, operating in a non-hostile environment, cannot be subject to side-channel attacks). However, in order to prevent, or at least detect, DoS attacks, it is still necessary for the receiver to send an acknowledge (or failure) message back. This is performed using the symmetric AE, as described in the one-to-one process detailed above.

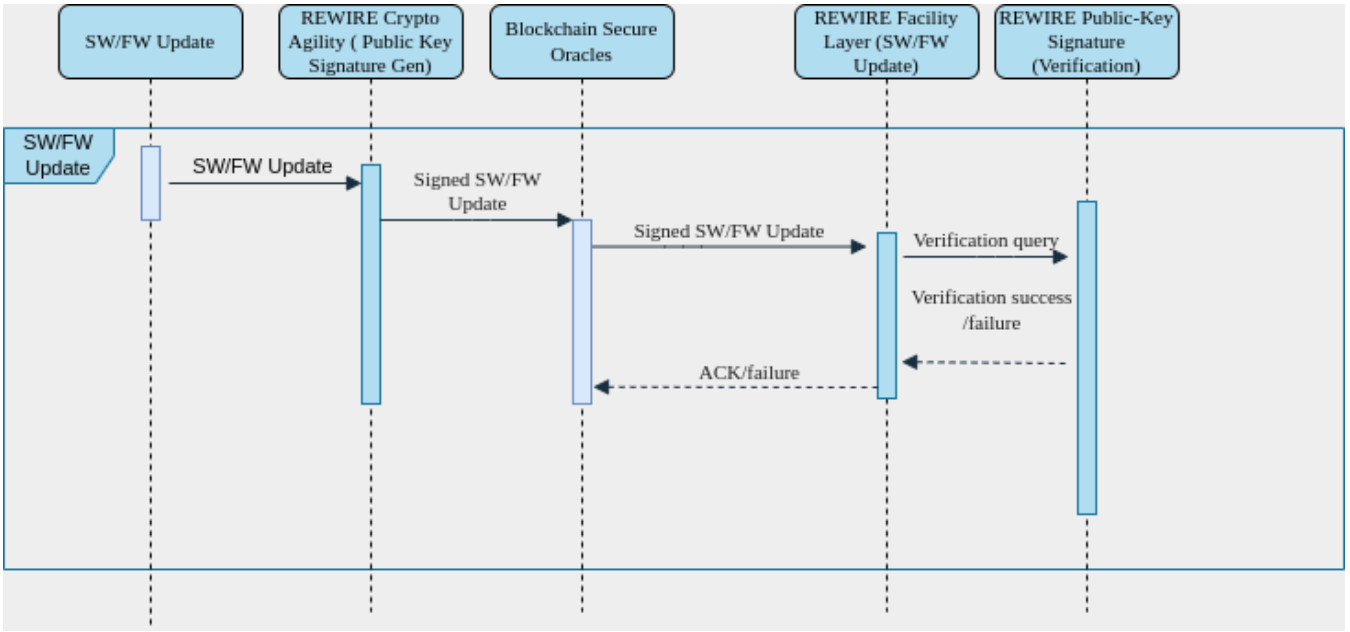

Figure 4.4: SW/FW update using AE (one-to-many)

4.3.3. SW and HW co-design Formal Verification and tools

4.3.3.1. Description and functional objective

SW and HW co-design of a system using integrated formal verification tools is considered an advanced engineering process of using the right tools and methodologies for generating evidence that the SW or HW artefacts meet its specifications. In this section the description of the model-based design framework for Validation and Verification (V&V) of SW or HW artefacts will be described including methods and modelling approaches for a) mapping requirements to artefacts, b) verifying functional and requirements-driven correctness and c) validating security requirements through evidence wrapped in assurance cases. Model-based Engineering (MBE) has emerged as a key set of methodologies to design complex systems. One widely adopted MBE technology is the AADL [REF-01]. Initially developed for avionics applications, AADL has since been used to design a wide range of embedded real-time system architectures, largely due to its language constructs for specifying both software and hardware configurations. Moreover, AADL has a reference implementation called OSATE [REF-02], which is an open-source modelling environment that comes with a few built-in analysis tools such as flow control and schedulability. Because OSATE is based on the Eclipse framework, creating new analysis plugins is relatively straightforward. AADL includes an annex mechanism for extending the base grammar, thereby supporting new language features and analyses. One such annex is the AGREE [REF-03], which is a compositional assume-guarantee-style formal analysis tool. AGREE attempts to prove properties about one layer of the architecture using properties allocated to subcomponents. The composition is performed in terms of formal assume-guarantee contracts that are provided for each component. Assumptions describe the expectations the component has on its inputs and the environment, while guarantees describe bounds on the component's behaviour. The model checker then attempts to find any model execution traces that violate these contracts using one of several SMT solvers. If the model checker covers all reachable states in the model without finding a violation, the model is proven to satisfy its contracts.

AADL is a standardized architecture description language, able to model hierarchical structure and connections among software and hardware components. The language was designed with extensibility in mind, which can be done in two ways: (a) custom component properties and data types, as well as (b) In the context of REWIRE, AADL is intended to be used in order to model a repository of proof artifacts for the various (security) properties related to the project use cases. In particular, the Resolute annex will be used to collect the available evidence from the various model checking and theorem proving tools and build an assurance case stating that the top-level system operates securely.

Representing requirements in AADL can be done using custom data types and component properties. In particular, a record type can be used to collect relevant information about a specific requirement (such as textual description, ID, path to the disk where proof artifact can be found, etc.), and then custom properties of this requirement data type can be associated with the various system components in order to map requirements to the system architecture.

The Resolute annex can be used to collect information about requirement satisfaction in an AADL model, by checking the filesystem for existence of the relevant proof artefacts. If all required artifacts are present, the tool outputs that the top-level system is secure. Otherwise, it pinpoints to the particular component from which the relevant proof artefacts are missing and need to be provided.

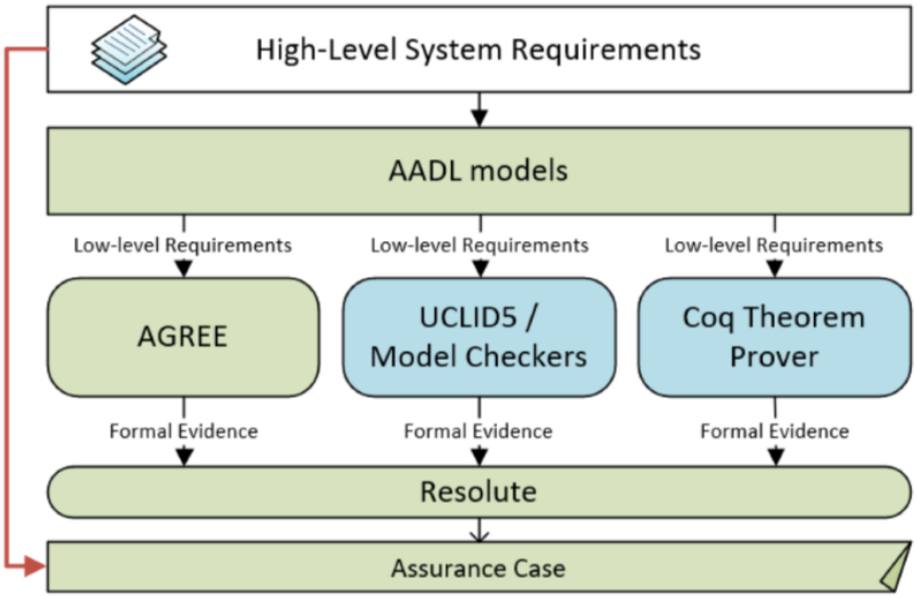

Figure 4.5: High level overview of validation and verification REWIRE framework

Figure 4.5 highlights the baseline V&V workflow that we envision as part of REWIRE. Starting from high level system requirements mapped in AADL models (architecture) and tracing the requirements within evidence generated by specialized formal verification tools (Model checkers, Theorem provers). Evidence will be linked to each individual requirement through the RESOLUTE tool validating (or not) each assurance case that described the requirement.

The framework is intended to be used according to the following workflow:

1. Security requirements are specified in order to mechanise them in the framework.
2. System architecture is modelled in AADL by defining key components.
3. Formal analysis of model is performed using AGREE to verify the design satisfies security properties captured in functional requirements.
4. Hardware and Software components are implemented manually, or through verified synthesis and linked in the AADL components.
5. Where possible, formal analysis is performed on component implementations – this could be done by a variety of methods (e.g., model checking and theorem proving). Requirements are being modelled as Temporal specification or theorems within each approach.
6. Component implementations are integrated into a system build.
7. System V&V is performed – checking the link between the generated evidence and the requirements associated with it.
8. An assurance case is generated using Resolute, confirming that security goals are supported by evidence (maintained by the framework).

4.3.3.2. Component workflows

Three are the main challenges model checking has to overcome. First, both the system and the property need to be amenable to formalisation. Second, the existence of a decision algorithm is determined by the language used in formalisation; it might be possible to employ algorithms that are either capable of proving validity or showing invalidity, but not both. Third, such algorithms need to be able to scale to complex systems, where the number of system states grows exponentially with parameters and components. This problem, known as “state space explosion”, can be addressed by means of techniques like abstraction, which simplifies the model to an overapproximation of it, so that if the property holds in the overapproximation, then it holds for the original model; this comes at the cost of spurious counterexamples, only applicable to the abstract model, which need to be identified and removed.

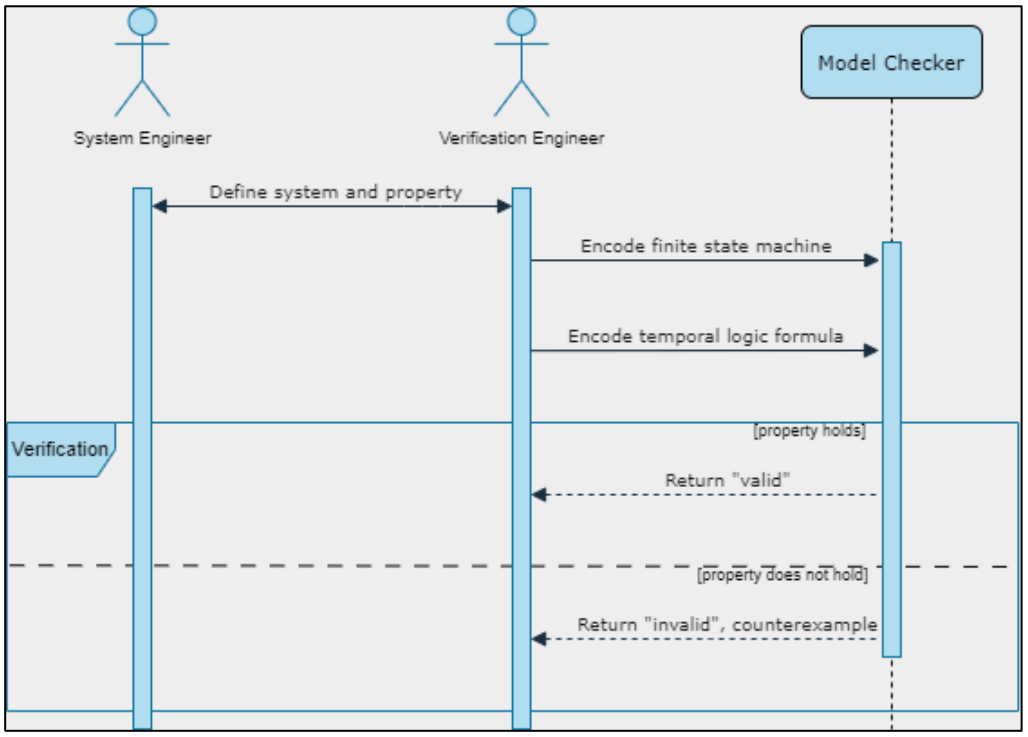

Figure 4.6: Model checking sequence diagram

Theorem proving is typically done with the help of proof assistant software (such as Coq, Isabelle, F*, Lean). It can in principle handle larger systems and more types of properties than model checking can, however it also typically requires user guidance (not automated), although many theorem provers include ways to automate trivial proofs.

In the context of REWIRE, theorem proving will be used to prove functional correctness of (parts of) the AES mode of operation developed by UCL. Requirements in theorem proving are represented as the top-level goal (or set of goals) to be proven and are typically expressed in some form of logic (e.g., propositional, first order, higher order). It should be noted that if the properties / theorems to be proven refer to particular aspects of a model, these aspects need to be modelled as well in the language of the theorem prover, which typically resembles a functional programming language (i.e., using recursion instead of loops, featuring immutable data structures etc.

Using a proof assistant typically requires manual effort. After modelling the requirements as the high-level goal to be proven, if this goal is simple enough, the tool can prove it automatically. Otherwise, the user has to come up with a way to decompose the high-level goal into subgoals. Proof that a goal is implied by its subgoals is typically trivial (done automatically by the tool). Then, the user has to guide the tool into proving the subgoals. These can be trivial or not, in which case they may need to be broken down into further subgoals, until everything can, eventually, be verified by the tool.

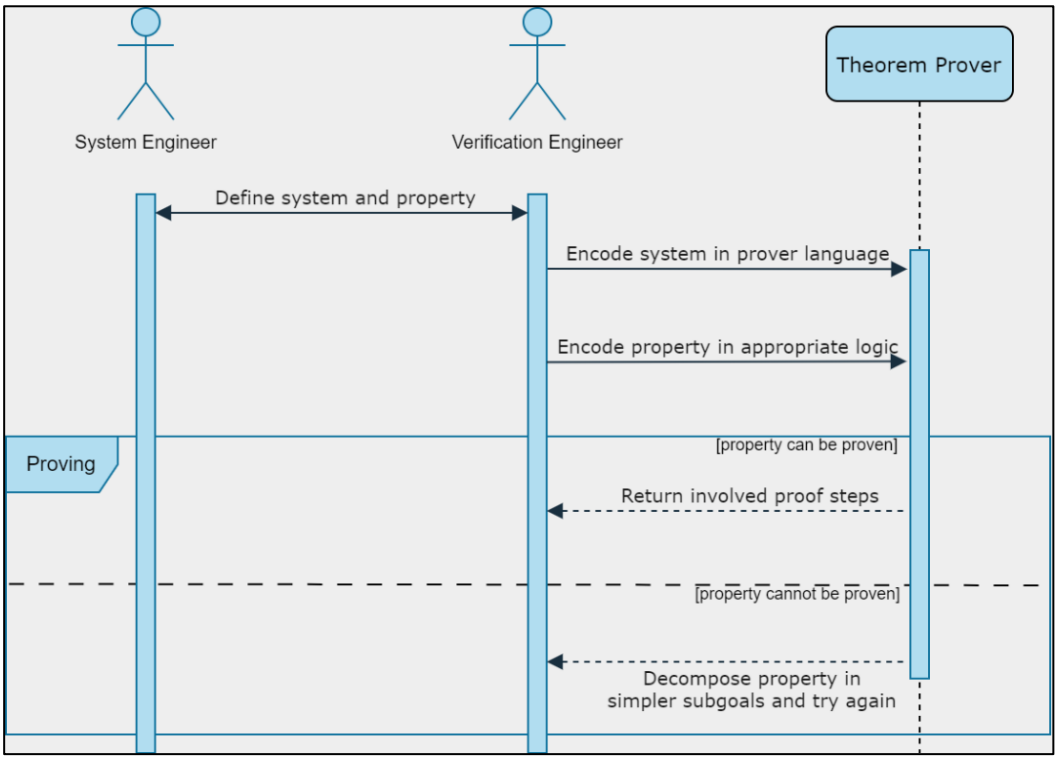

Figure 4.7: Theorem proving workflow executed at design time

Model Checking Example

As a simple example to understand how model checking works, let us consider a function that adds two 8-bit unsigned integers a and b, both constrained to take values smaller or equal than 128. The objective is to verify whether unsigned overflow is possible under these conditions. Using the formal modelling language *SMV*, the adder can be represented as follows:

```
FROZENVAR
 a : unsigned word [8];
 b : unsigned word [8];
INVAR (a <= 0ud8_128);
INVAR (b <= 0ud8_128);
DEFINE
 c := a + b;

INVARSPEC (c >= a & c >= b);
```

The inputs a and b are encoded as unsigned words 8 bits long, constrained as needed; the output *c* is defined as the sum of a and b; the target property states that c must be greater or equal than both a and b.

An execution of the model checker *nuXmv*, that uses SMV as input language, shows that the property is not valid and that indeed unsigned overflow is possible, by returning $a = 128, b = 128$ as a counterexample assignment, in correspondence of which $c = 0$.

```
-- Property (c >= a & c >= b)  is false
-- as demonstrated by the following execution sequence
Trace Description: Incremental COI counter-example
Trace Type: Counterexample
  -> State: 2.1 <-
    a = 0ud8_128
    b = 0ud8_128
    c = 0ud8_0
```

4.3.4. Secure System configurations and Security Controls Enforcement

4.3.4.1. Description and functional objective

The REWIRE architecture will focus on mechanizing the formal verification outputs into enforceable security policies that will govern the device operation throughout the executed use-case scenarios. This attempt will be a semi-automated activity as the verification results extracted from specialized models at the design time will require the formal verification expert to interpret the outcome into tangible security constraints, to meet the original requirements, proven from the models but not yet realizable into the device. As in the current state of the art, formal verification processes are often decoupled from the software or hardware artefacts development (unless there is a sound code-HDL generation module), it is of paramount importance for the security engineers to follow the models' blueprints (e.g. inter-outer procedural calls, sequence of execution, atomic operations or deterministic algorithmic steps) in order to 'secure' their implementations from property violations previously proven using formal verification tools.

4.3.4.2. Component workflows

From the REWIRE architecture point of view, policy synthesis and enforcement will be composed by two different engineering operations. Those operations will be mainly driven by the outcome of the formal verification approaches (successful or fail result), enforcing the engineer to attempt further safeguarding routines to be added on top of the device. Those routines will be following either the form of monitors or conditional constraints-rules that will protect the device against events previously not proven by formal verification techniques or from properties for which the verification has generated detailed counterexamples. Thus, REWIRE security controls will be categorized as follows:

- **Formally verified design artefacts**: for a series of critical components' interconnections residing either at software or hardware level, after the successful correctness verification, the underlying assurance cases (models connected to evidence proving their properties) will enforce the presence of the necessary links (e.g., inputs/outputs from inner-device communication) between different components. The latter, based on application-driven requirements may be mapped both at the trusted or the untrusted zone of the device. Formal verification will guaranty that the presence of the link is necessary; at the same time, a security enforcement rule may enforce the establishment of the link overriding any authorization principles that from the application level is it required to be executed. Links could be also present and validated between the DSP cores (executing software tasks) and the RISC-V hardware accelerators that perform critical cryptographic operations per user request. If for example, isolation is proven to be correctly implemented through formal verification, and if the representative isolation model is being followed to architect and develop the isolated core, then there is no need for the policy to form additional constraints to safeguard the core's isolation since this is proven by-design. If though a new extension of the core is being considered, without capturing this at the model level, then the security controls with respect to core isolation properties need to include additional checks at the core level, whether e.g., an atomic execution of critical processes remain uninterrupted from other events. Finally, failed verification outcomes that also provide evidence of property violation can also yield counterexamples (state-based information) that the user may visit to identify the error and correct it if possible. If for some engineering issues the property remains violated, then the counterexample could be used as a 'negation' monitor that will prevent the execution of events that may lead to a real state where the property is violated.
- **Security policy optimization and management**: The security policy synthesis and optimization will consist of a) constraints derived from the formal verification outcomes, b) additional security controls derived by requirements not captured in verification models and c) not verified requirements that are being considered as assumptions on the verification models. All documented assumptions that have been defined at the formal verification activities need to generate a list of constraints to be enforced by the policy; those assumptions based on expert knowledge review, can yield additional constraints to the device operation level that couldn't be verified at the design time, yet it is necessary to be validated at run-time. At this stage an optimized and conflict resolution mechanism will be also in place to effectively generate the ideal

enforcement rules that do not overlap or conflict with each other.

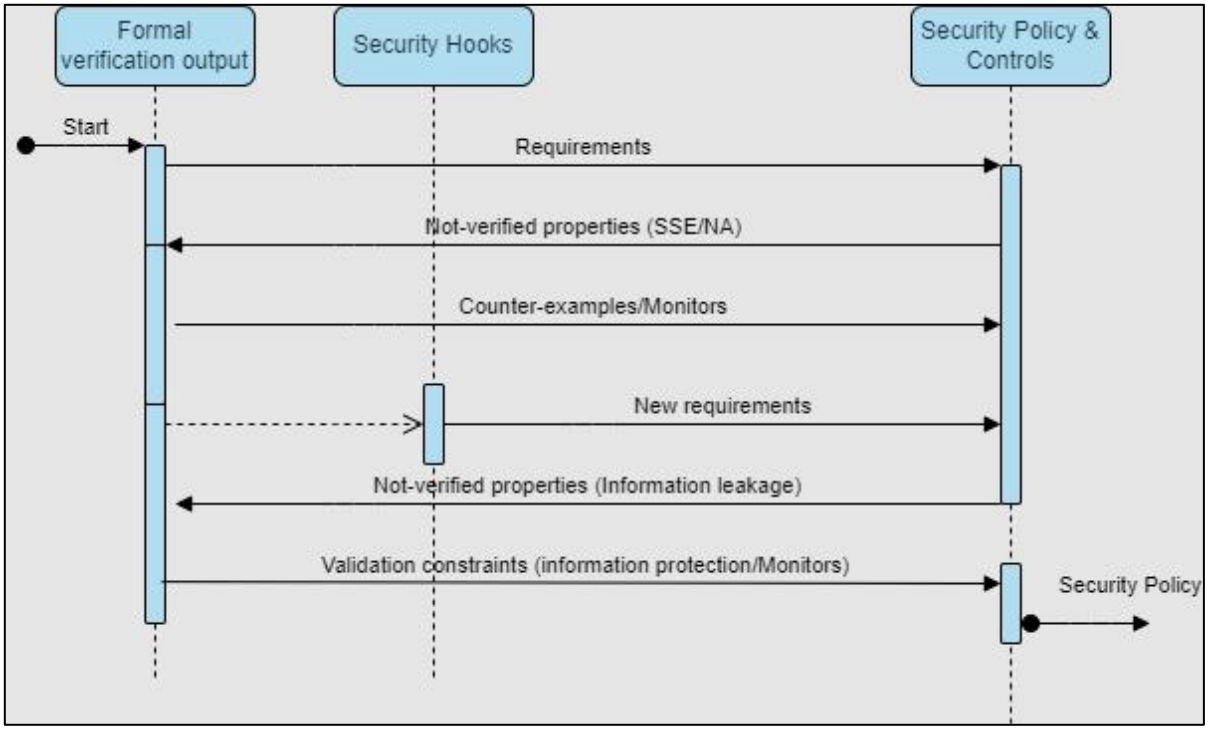

Figure 4.8: Security Policy & Controls sequence diagram

4.4 Runtime Architecture and Functional Components

While the previous section was focused on the description of the components comprising the design-time phase, this section focusses on the individual components of the runtime phase of REWIRE. For each of them, details of their design and their functional objectives are given.

4.4.1. Risk Assessment

4.4.1.1. Description and functional objective

The main objective of the risk assessment, as its name suggests, is to perform real time monitoring and evaluation of the potential risks on the underline devices, so that to enable the Security Administrator to take informed decisions, corrective actions or re-evaluate the theorem proofs of the design and runtime security measures. Risk assessment (RA) is part of both the Design and Runtime phases of the deployments' lifecycle. RA provides real-time monitoring and evaluation of the a) Risk Indicators provided from the AI-based Threat Intelligence and b) the SW/FW Validation components and c) the previous calculated Risks (during design time or the last executed risk assessment during runtime) in order to generate the (updated) risk graph and (re-)calculate the cyber security risks, tailored to the REWIRE's IoT ecosystem. The Risk Assessment is a core part of REWIRE and provides qualitative and quantitative risk estimations to increase cybersecurity awareness.

4.4.1.2. Component workflows

In the context of RA, two main workflows have been identified. The workflows include the Risk Assessment, the SW/FW Validation, and the AI-based Threat Intelligence components independently of the running phase (e.g., design or runtime). Both workflows trigger the Risk Assessment based on the potential identified Risk Indicators either from the AI-based Threat Intelligence or the attestation results

produced by the SW/FW Validation and generate the new calculated Risks. Below we provide in detail the workflows in both phases. The risk assessment component is used mainly in the context of the runtime phase for dynamically analysing the risk indicators stemming from the various detection/monitoring. However, in this section we refer also to the design time phase, as the risk assessment can be used to perform an initial analysis of the risks existing in an initial deployment and can be considered in the design time phase.

Design Time Phase: During this phase the Risk Assessment component runs for the first time. Thus, no previous risk calculations will be considered. The Security Administration provides for the first time the underline Assets, the Requirements, and the Security Policies against which the Risk Assessment will calculate the Risks. The Risk Assessment component considers the asset topology and the underline threats/vulnerabilities so that to generate the risk graph along with the corresponding risks.

Runtime Phase: During this phase, the Risk Assessment component runs after the first time. Thus, previous risk calculations will be considered for the new risk estimation. The Security Administration reevaluates potential updates on the Assets, the Requirements, and the Security Policies against which the Risk Assessment will calculate the Risks. In this phase, when the AI-based Threat Intelligence detects a misbehaviour, or the SW/FW Validation a failed attestation a new Risk Assessment process is initiated. Now the Risk Assessment also considers, apart from the asset topology, the threats/vulnerabilities and security events, the previously generated risk graph.

Figure 4.9 below depicts these two flows that initiate a new risk assessment during runtime.

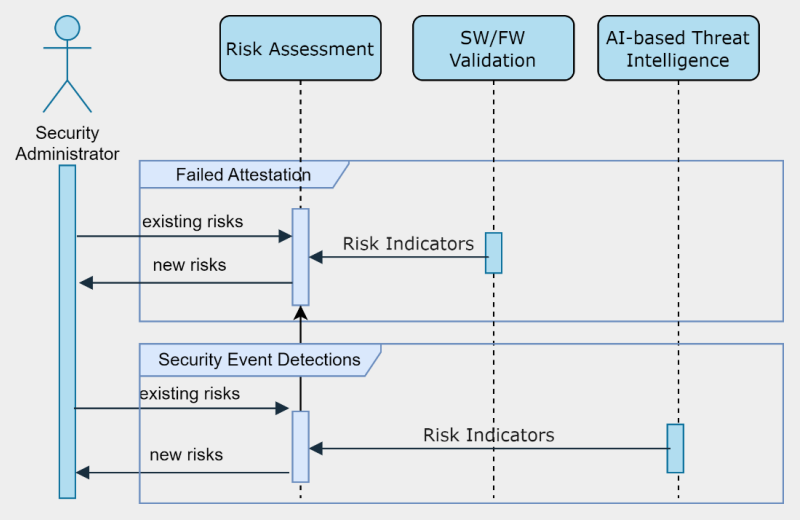

Figure 4.9: Risk Assessment Component Workflow

4.4.2. AI-based Threat Intelligence

4.4.2.1. Description and functional objective

The AI-based Threat Intelligence engine is a hybrid local cloud-based component designed to analyse and detect threats that may arise during the REWIRE Runtime phase. It leverages the power of Artificial Intelligence (AI) algorithms trained on vast amount of data. The training of the algorithm(s) utilises the data collected through the Secure Oracles and stored in the Off-chain Data Storage and is performed over the cloud-based infrastructure of the engine. Upon training of the algorithm(s), the AI-based Threat Intelligence is capable of undertaking automatically on-the-fly misbehaviour detection, by utilising the data ingested through the Secure Oracles, where part of the operations is performed at the Secure Oracle level (for example the data-veracity activities), and other at the cloud level. The key output of the AI-based Threat Intelligence will be values of risk indicators related to the identified threats, that will be provided as input to the Risk Assessment module so that the System Administrator can identify the threats and take

promptly corrective actions. In this direction the primary objective of the AI-based Threat Intelligence engine is to detect and identify potential threats, by analysing a wide range of data, including network logs, system logs, etc., from the project's pilots to identify suspicious activities or patterns that may indicate an impending threat. To achieve this the engine uses a large volume of historical data to train AI algorithms (manual process); thereafter the trained AI algorithms will continuously analyse (automated process, based on defined schedules) incoming data streams in near real-time, thus enabling the engine to detect emerging threats promptly. By leveraging the power of AI, the engine can handle a vast amount of data quickly and efficiently, ensuring a high-level of accuracy in threat detection. Moreover, the engine will continuously learn and adapt to evolving threat landscapes, by incorporating feedback loops to update and refine its algorithms based on new data and emerging threat patterns. This adaptive learning capability enhances the engine's effectiveness in detecting emerging threats, providing robust and efficient threat intelligence capabilities with the REWIRE ecosystem.

4.4.2.2. Component workflows

As shown in the following Figure 4.10, two key workflows are anticipated for the operation of the AI-based Threat Intelligence engine within the REWIRE framework.

The first one entails the *training of the AI Misbehaviour detection algorithm(s*), where transactions (i.e., data request) are made from the AI-based Threat Intelligence engine to the Secure Oracle, which in turn requests the appropriate data location pointers from the Blockchain Infrastructure. Utilising these pointers, the Secure oracle can fetch the required data files from the Off-Chain Data Storage, which in turn returns the data files related to the pointers. This process ensures that sensitive data remains protected and is only accessible by authorized entities. Finally, having acquired the data files manual training of the AI Misbehaviour detection algorithm can take place within the engine, however the acquired data files may undergo preprocessing such as cleaning, normalising, and transforming the data into a suitable format for the training process. If necessary, adjustments and improvements are made to enhance the model's accuracy and efficiency. After successful training and evaluation, the AI Misbehaviour detection algorithm is deployed in the AI-based Threat Intelligence Engine's operational environment. To ensure the algorithm's ongoing effectiveness, scheduled feedback loops are anticipated to be established to collect new data for periodically retrain and update the AI Misbehaviour detection model, ensuring it stays up to date with emerging threats.

The second workflow involves the *near-real time Misbehaviour detection*: Upon training of the algorithms, the AI-based Threat Intelligence Engine is capable to undertake "on-the-fly misbehaviour detection", automatically (based on predefined schedules) executing the model to analyse incoming data streams in near real-time, enabling the detection of emerging threats promptly. As a starting point, the AI-based Threat Intelligence Engine registers with the Secure Oracle, which in turn checks the veracity of the data requested and pushes data to the Off-*Chain Data* Storage. Once satisfied, the Blockchain Infrastructure returns the appropriate data pointers. Each data request is interpreted as a new transaction, recorded in the Blockchain Infrastructure which sends back to the Secure Oracle the relevant transaction pointer, so it can proceed with requesting the data from the Off-Chain Data Storage. The data are returned to the Secure Oracle, and thereafter pushed back to the AI-based Threat Intelligence Engine, which executes the model towards identifying threats. In the event abnormalities or threats are identified from the model, the AI-based Threat Intelligence Engine will push values of risk indicators to the Risk Assessment component.

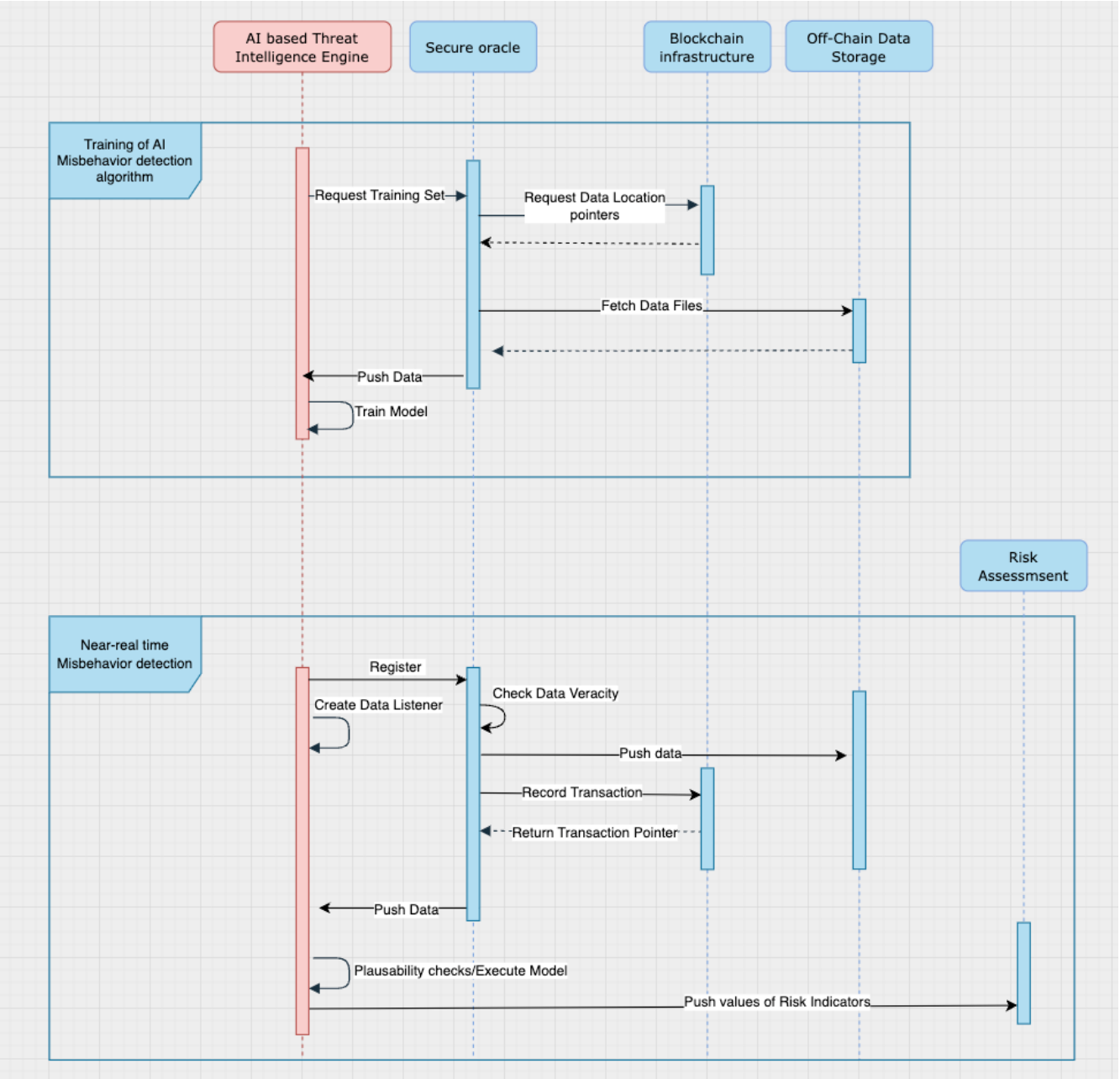

Figure 4.10: AI-based Threat Intelligence Workflow

4.4.3. Blockchain Infrastructure

4.4.3.1. Description and functional objective

The Blockchain Infrastructure serves as the foundational and robust framework within the REWIRE project, providing a decentralized and secure platform for trusted data verification and access control for all users. It functions as an immutable and distributed ledger, storing a tamper-resistant record of transactions and data interactions within the system. The primary objective of the Blockchain Infrastructure is to establish a transparent and verifiable data management system, ensuring the integrity and reliability of data exchanged and accessed by various components within the REWIRE ecosystem. Through the use of blockchain technology, the component ensures that all data interactions and transactions are recorded in a chronological and transparent manner. Each data entry is securely linked to the preceding one, creating an unbroken chain of data history that cannot be altered or modified retroactively. This inherent immutability safeguards against unauthorized tampering and enhances the overall trustworthiness of the system. Moreover, the decentralized nature of the blockchain ensures that no single entity has full control over the data, preventing any potential single point of failure and enhancing

the system's resilience and fault tolerance.

Another pivotal aspect of the Blockchain Infrastructure is its role in managing access control for users within the REWIRE ecosystem. Utilizing smart contracts and cryptographic mechanisms, the blockchain enforces predefined rules and permissions, allowing users to interact with specific data and functionalities based on their authorization level. This ensures that sensitive data is only accessible to authorized personnel, bolstering data privacy and security. By providing a transparent, tamper-resistant, and decentralized data management system with robust access control mechanisms, the Blockchain Infrastructure component enables the REWIRE project to achieve a high level of trust, accountability, and efficiency in handling critical data and transactions. Smart contracts are a fundamental component of the REWIRE project's Blockchain Infrastructure. REWIRE generally utilizes two smart contracts: the Oracle Contract and the User Contract. The oracle contract serves as an interface to off-chain oracle components that allows users to request external/internal data and handle data delivery, while the user contract interacts with the main oracle contract to manage user identity and access control and handles request response and cancelation. These two contracts together enable secure and decentralized data retrieval and processing within the REWIRE blockchain ecosystem.

4.4.3.2. Component workflows

In the REWIRE ecosystem, the Blockchain Infrastructure emerges as the central, wielding its power to facilitate data flow and enforce crucial access control and smart contract functions. It is important to note that although the blockchain is a closed and trusted system, incoming external data sources rely on another component. This is where the Secure Oracles component takes the stage, engaging in a sequence of actions that begin with data verification requests from External Data Sources. Ensuring security, the Secure Oracles relies on TEEs to process data within isolated enclaves, guaranteeing confidentiality and integrity. The integration of a TEE into the Secure Oracle is crucial because it allows oracle's critical operations, such as data retrieval and data veracity check, to occur in a secure enclave, shielding them from potential interference by malicious processes or the underlying operating system. This isolation ensures the confidentiality and integrity of data. For instance, consider a scenario where the oracle fetches stock market data for a decentralized trading platform. Without a TEE, malicious software on the host machine could manipulate or eavesdrop on this sensitive data. However, with a TEE, the data retrieval process is protected, and oracle can cryptographically prove to remote clients that it operates within a secure enclave, guaranteeing the authenticity and trustworthiness of the data.

Upon successfully executing a specific oracle operation, typically cantered on data retrieval from external sources, there are two attestation processes that come into play within the TEE. The first pertains to the validation of the data's origin, ensuring that the source of the data maintains its integrity and authenticity. This is imperative, as it vouches for the purity and legitimacy of the data before any further action is taken. The second type of attestation dives deeper, focusing on the secure oracle itself. This attestation is crafted to guarantee that the state and condition of the oracle remain uncompromised and that its operations are conducted in a secure and trustworthy manner. In essence, these dual attestations not only validate the data's origin but also secure the very processes of the oracle, establishing a robust shield of trust and security. Subsequently, both the processed data and its corresponding attestation are securely stored within a blockchain-based smart contract, forming an immutable record of the operation's validity and the data's reliability within the REWIRE ecosystem. This decentralized and immutable ledger safeguards the data, ensuring permanence, transparency, and tamper resistance, underlining the ecosystem's unwavering trustworthiness and data reliability.

Moreover, in cases where data surpasses Blockchain capacity, Off-Chain Data Storage acts as a reliable alternative, harmonizing data and indexes seamlessly. It's pivotal to understand that when data is submitted to the Off-Chain Data Storage, it is often processed with suitable encryption or signing primitives, depending on the requirements. Typically, data is stored encrypted, and a pointer to this encrypted data is then saved to the blockchain. Exclusively relying on the secure Oracle as the singular entry point for data transfer into the blockchain, REWIRE maintains a distinct approach compared to other blockchain systems where peers might have the capability to upload data. This strategic decision ensures a stringent security framework, as even data uploaded by internal entities within REWIRE undergoes meticulous filtration and verification processes by the Oracle. This rigorous protocol is imperative to

uphold the highest standards of security, making certain that all data entering the blockchain aligns with REWIRE's uncompromising safety measures. The aforementioned workflow is depicted in Figure 4.11 below.

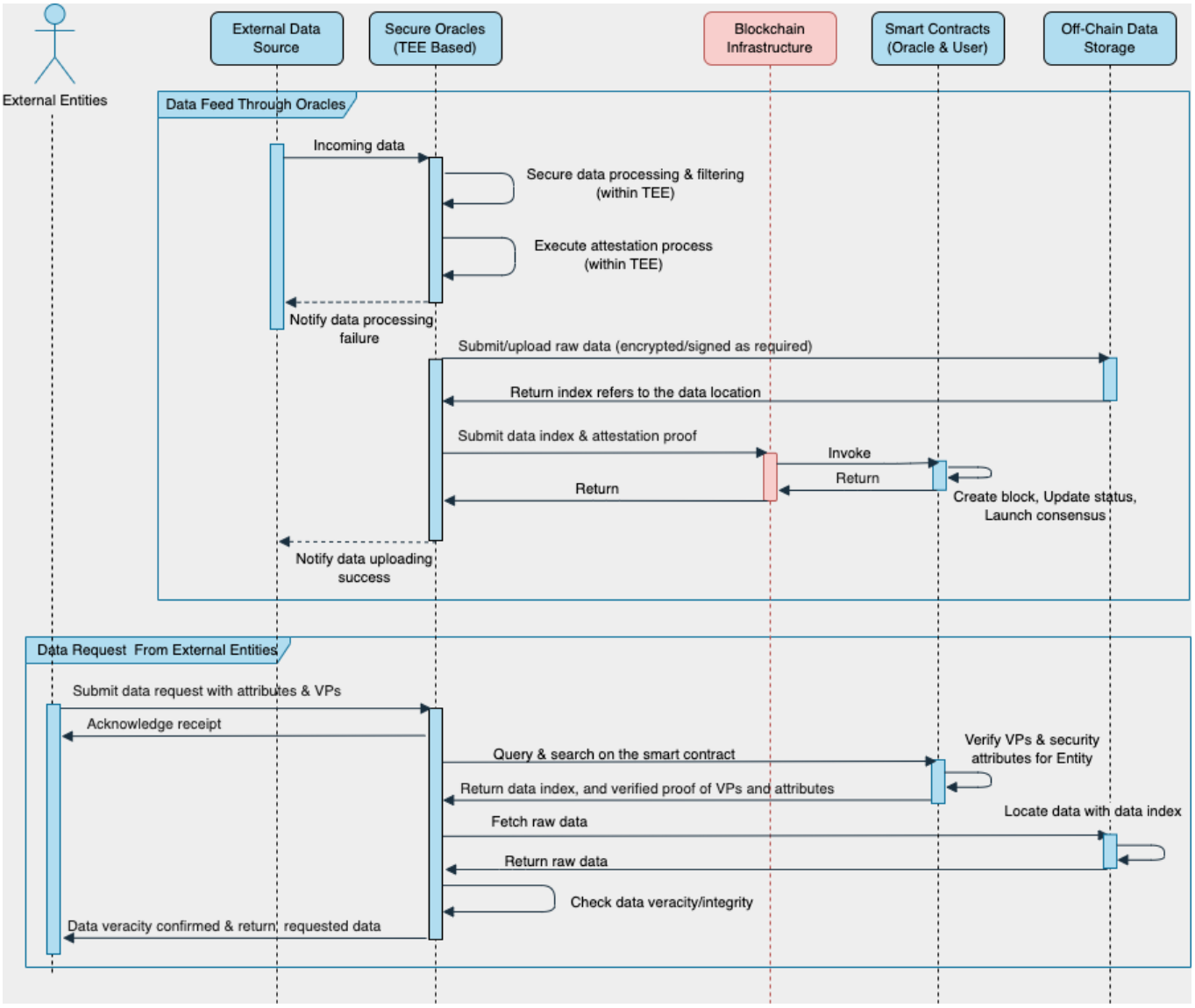

Figure 4.11: Blockchain Infrastructure Workflow

ABAC-based data sharing

The REWIRE project's Blockchain Infrastructure introduces a dynamic layer of data querying and pushing, underpinned by the Attribute-Based Access Control (ABAC) mechanism. Beyond standard access systems, ABAC offers a context-rich approach, factoring in attributes of the user, the resource, and the environment. Within the REWIRE framework, these attributes are closely tied with Verifiable Presentations (VPs) and specific security-related attributes that devices manifest. Such granularity in data interaction ensures that not only is data access transparent but also inherently secure.

When entities, whether users or devices, intend to query or push data to the blockchain, the journey of their request starts with the oracles. Acting as gatekeepers, oracles first assess the request's attributes and VPs. These VPs, complemented by security-related attributes like device trustworthiness score or prior interaction logs, are pivotal in discerning the authenticity and intentions of the requesting entity. After the oracles validate the entity's attributes and VPs, they perform another crucial task: verifying the veracity of the data. This step is non-negotiable, ensuring that the data being interacted with maintains its integrity and has not been compromised. Such validation becomes especially vital when the data is instrumental, like in cases where it feeds an AI-based misbehaviour detection system. Only after the oracles have

successfully vetted both the entity's credentials and the data's veracity do the request proceed to the Blockchain for transaction completion.

BC-based Distribution of SW updates

In the ever-evolving digital landscape, software (SW) updates are indispensable to ensure the security, efficiency, and relevance of systems. Leveraging the REWIRE project's Blockchain Infrastructure for SW update distribution brings transparency, accountability, and ensures the authenticity of distributed updates. Oracles, as intermediaries between the blockchain and external entities, play a pivotal role in managing and verifying these updates.

The REWIRE framework's software update journey commences with the Software Service Provider signing and encrypting the update. This process results in a reference value, usually a hash or digital signature, which captures the update's essence. Alongside pertinent metadata, such as version specifics and targeted devices, this value gets a permanent spot on the blockchain via the Secure Oracle. Concurrently, the actual software update is stored in off-chain storage for secure and efficient retrieval. Devices, when alerted about new updates by the Oracle, don't directly access the blockchain or off-chain storage. Instead, the Secure Oracle serves as their primary gateway. Upon a device's request for an update, the Oracle first consults the blockchain to fetch the reference value. Subsequently, it interacts with the off-chain storage to retrieve the actual software update, ensuring the update's sanctity remains uncompromised. Upon receiving the update and its reference value from the Oracle, devices have the autonomy to assess the update's integrity. They use the reference value as a validation benchmark, ensuring the update's authenticity. After successful validation, devices integrate the update, marking the completion of its journey from a blockchain record to a tangible real-world implementation.

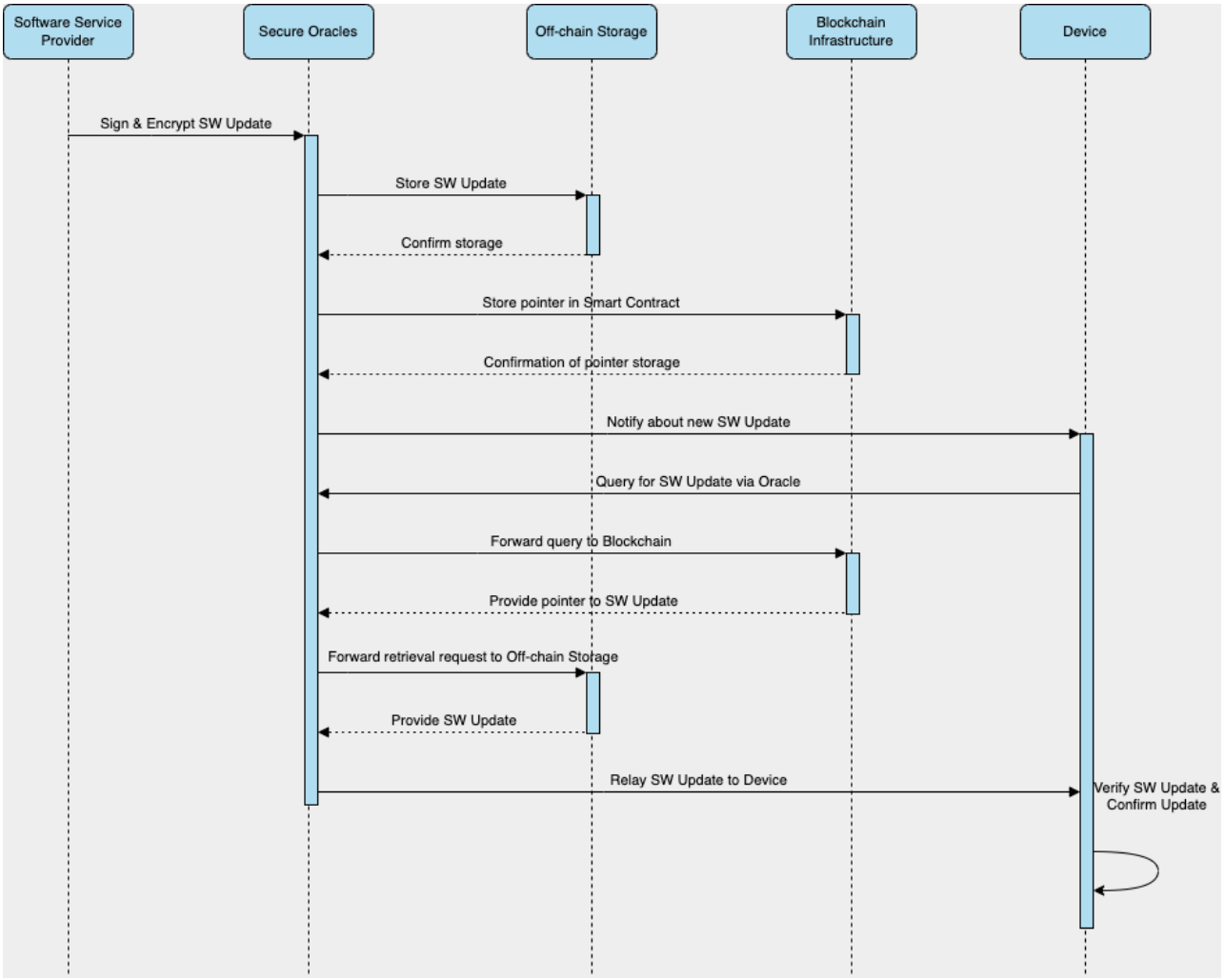

Figure 4.12: Blockchain Based Distribution of SW Updates

4.4.4. Secure Oracles

4.4.4.1. Description and functional objective

Secure Oracles serves as the pivotal and trustworthy bridge that seamlessly connects the blockchain infrastructure to external data sources residing in the off-chain realm within the REWIRE ecosystem. It plays a critical role in facilitating secure and reliable data ingestion, enabling the blockchain to interact with real-world information while ensuring data integrity and confidentiality. The primary objective of Secure Oracles is to establish a resilient and tamper-resistant communication channel that allows the system to access and utilize valuable data from off-chain sources without compromising the security and immutability of the underlying blockchain. The component leverages the power of TEEs, a state-of-the-art technology designed to create secure enclaves for executing sensitive operations, guaranteeing that data exchanged between the blockchain, and off-chain sources remain protected from unauthorized access and tampering. By harnessing the capabilities of TEE, Secure Oracles establishes a secure layer of communication that safeguards against potential threats and ensures the veracity and authenticity of the data being ingested into the system. The use of TEE-based technology adds an extra layer of assurance that data veracity activities are performed with utmost reliability, boosting the overall trustworthiness and credibility of the REWIRE ecosystem.

Furthermore, Secure Oracles operates with the goal of enabling smooth and seamless integration of external data into the blockchain environment. By acting as a trusted intermediary, it allows the AI-based Threat Intelligence engine and other components within the system to access a wide array of data sources, including network logs, system logs, and other relevant datasets. Overall, the Secure Oracles (TEE based) component plays a pivotal role in maintaining the integrity, security, and efficiency of data communication between the blockchain and off-chain data sources, contributing to the robustness and effectiveness of the REWIRE ecosystem's threat intelligence capabilities.

4.4.4.2. Component workflows

The presented sequence diagram illustrates the data flow and processing steps of Secure Oracles component in the REWIRE ecosystem. The sequence starts with the External Data Source sending data to the Secure Oracles for verification. At this juncture, Secure Oracles initiates a data verification Request, signalling its intention to securely process the received data. Within the realm of secure data processing, Secure Oracles engages TEE. This mechanism ensures that the data undergoes a series of secure processing steps within isolated enclaves, guaranteeing its confidentiality and integrity. Secure Oracles play a pivotal role in ensuring data veracity before it becomes part of a blockchain. Imagine a scenario where external financial transaction data is destined for blockchain inclusion. Oracles initiate a comprehensive verification process, encompassing not only integrity checks but also data format, version, and size validation. Initially, they create a unique data fingerprint using cryptographic hashing and compare it to a reference value on the blockchain to verify data integrity. Simultaneously, they validate the data's source through digital signatures. Once source authenticity is confirmed, an attestation is generated to affirm data authenticity and adherence to the specified format, version, and size. Following this, a second attestation is initiated, focusing specifically on the TEE or the secure oracle. This latter attestation assures the unblemished operational state of the oracle, underlining the robustness of the entire system. Only when the data successfully passes all these checks is it deemed eligible for blockchain inclusion. In the event that data processing fails, the Secure Oracles promptly notifies the Data Source about the processing failure, leading to the termination of the data processing procedure. As Secure Oracles successfully attests the secure execution of the data processing, the processed data and corresponding attestation are submitted to a Smart Contract in the Blockchain. Subsequently, the Smart Contract, acting as a decentralized and immutable ledger, stores both the processed data and its attestation securely within the "Blockchain. This storage within the Blockchain ensures a high level of data permanence, transparency, and tamper resistance, further solidifying the system's trustworthiness and data reliability. Finally, one optional process is that when the data to be uploaded to the blockchain is too large, the source needs to be dumped into the Off-Chain Data Storage, and the index of the data needs to correspond one-to-one with the storage location.

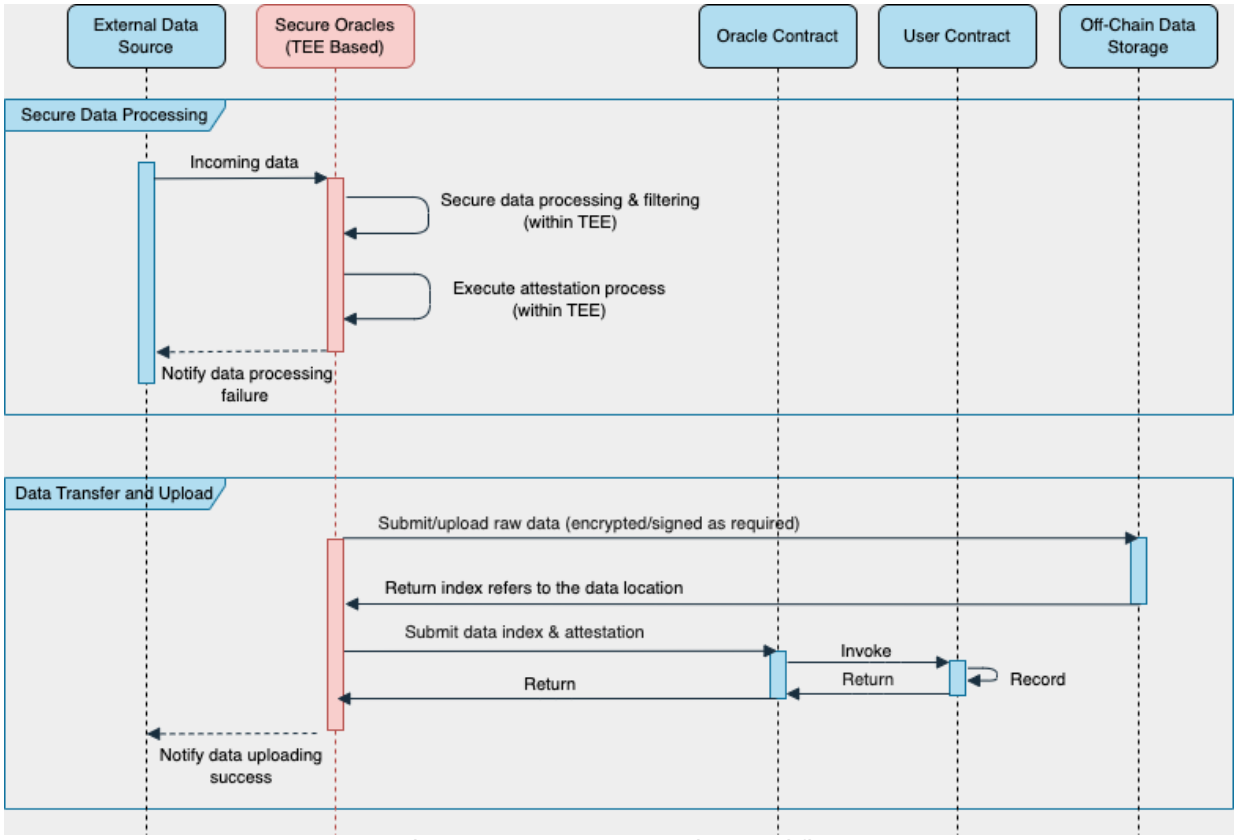

Figure 4.13: Secure Oracles Workflow

4.4.5. Off-chain data storage

4.4.5.1. Description and functional objective

The Off-chain Data Storage plays a crucial role in ensuring efficient and secure storage of data collected through the Secure Oracles to the Blockchain Infrastructure during the REWIRE runtime phase. Its primary purpose is to serve as a secure and dedicated data storage facility responsible for holding a wide range of data files generated from REWIRE runtime phase operations (such as attestation results and raw data from distributed sources) also serving as a repository for other sensitive information not intended for direct storage on the Blockchain Infrastructure. Within the Blockchain Infrastructure, the Off-chain Data Storage is referenced using pointers (indexes) that indicate specific data locations. When data is needed, requests are made by retrieving these pointers, effectively allowing access to the relevant data without exposing the actual content directly. By employing this mechanism, the Off-chain Data Storage maintains its security and confidentiality since its contents are not directly accessible by any other component or entity. Instead, all interactions with the Off-chain Data Storage are solely managed through the Blockchain Infrastructure, which ensures that only authorized operations, such as writing, updating, and reading data, are carried out, preserving the integrity and confidentiality of the stored information.

4.4.5.2. Component workflows

As shown in Figure 4.14, the data collected through the Secure Oracle, are stored in the Off-Chain Data Storage. Once the data are stored, indexing of the data takes place in Elastic, while the relevant data pointer(s) are sent back to the Secure Oracles maintaining them, so that it can be utilised in future data requests from other components (e.g., AI-based Threat Intelligence Engine). Data indexing involves organising and structuring the data in a way that makes it easily searchable and retrievable. This step enhances data accessibility and facilitates quick and efficient data retrieval when needed. Finally, data pointers are sent to the Blockchain Infrastructure updating also the transaction record. Overall, through this workflow, the system can efficiently collect, store, and retrieve data while maintaining a high level of security and confidentiality. The use of data pointers and the involvement of the Secure Oracle as an

intermediary enhances the overall integrity of the data storage and retrieval processes, making it a robust and reliable component within the REWIRE runtime framework.

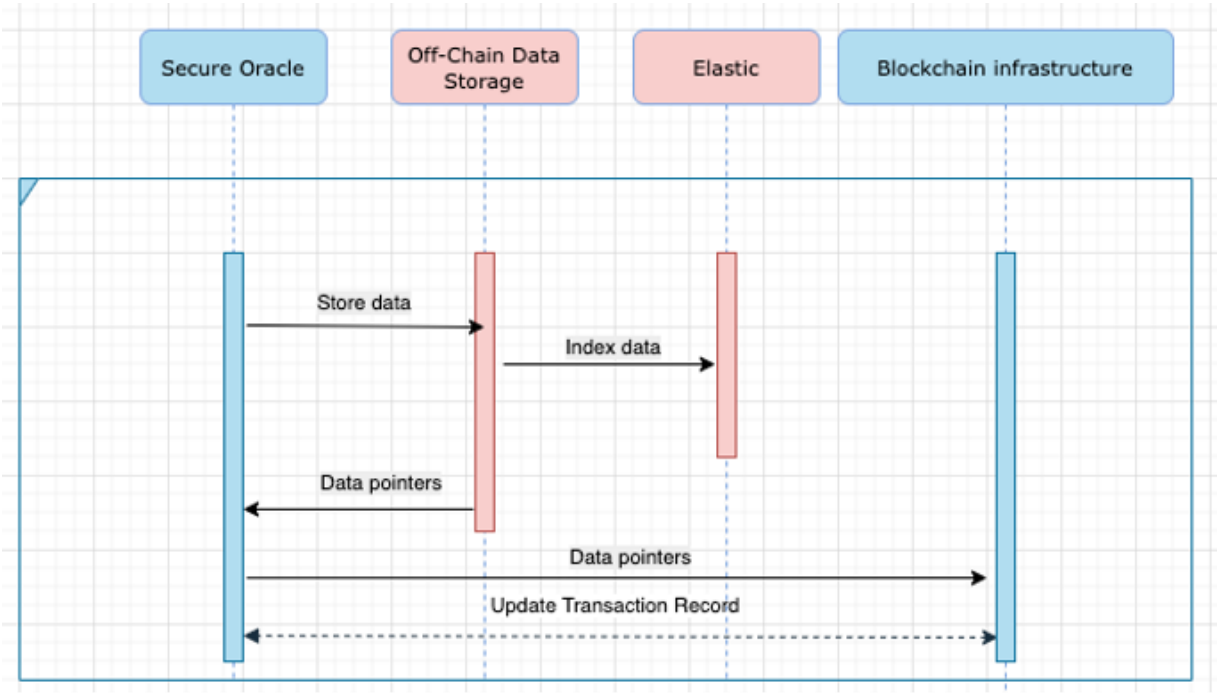

Figure 4.14: Off-chain Data Storage Workflow

4.4.6. Firmware & Software Validation

4.4.6.1. Description and functional objective

The primary objective of the SW/FW Validation component is to identify weaknesses in a given firmware/software image. It does so by using a combination of dynamic binary analysis by means of fuzzing, and static binary analysis by means of concolic execution. Thus, the information gathered on weaknesses is shared with various components of the REWIRE architecture. The secondary objective is to instrument critical software inside the firmware such that it can share information with other programs at runtime. An attestation agent running in the trusted environment of the REWIRE architecture will receive this information and use it to attest the correct functioning of the device at runtime. Any evidence resulting from this agent will then be shared to the blockchain.

4.4.6.2. Component workflows

During the REWIRE design time phase, the SW/FW Validation component will unpack the provided firmware image, taking out software that is deemed critical to the operation of the device. Using the file system found in the firmware, information about the device is gathered. The software is then emulated using this information and tested for weaknesses using its dynamic validation component and the static validation component. In case the dynamic component gets stuck during its analysis it will request additional information from the static component, which will be gathered using concolic execution. After the analysis finishes, its results are sent back to the user and to an instrumentation component. The instrumentation component will use this information to choose critical programs to instrument, as well as instructing the attestation agent on what correct behaviour of these programs looks like. After creating a firmware image with the instrumented programs and behavioural information it is deployed to a device. At runtime the instrumentation hooks in the programs will communicate with the attestation agent running in the trusted execution environment, which will attest the correct functioning of the program based on this information. Any results are then shared with the blockchain, which can distribute these results to other components in the REWIRE architecture.

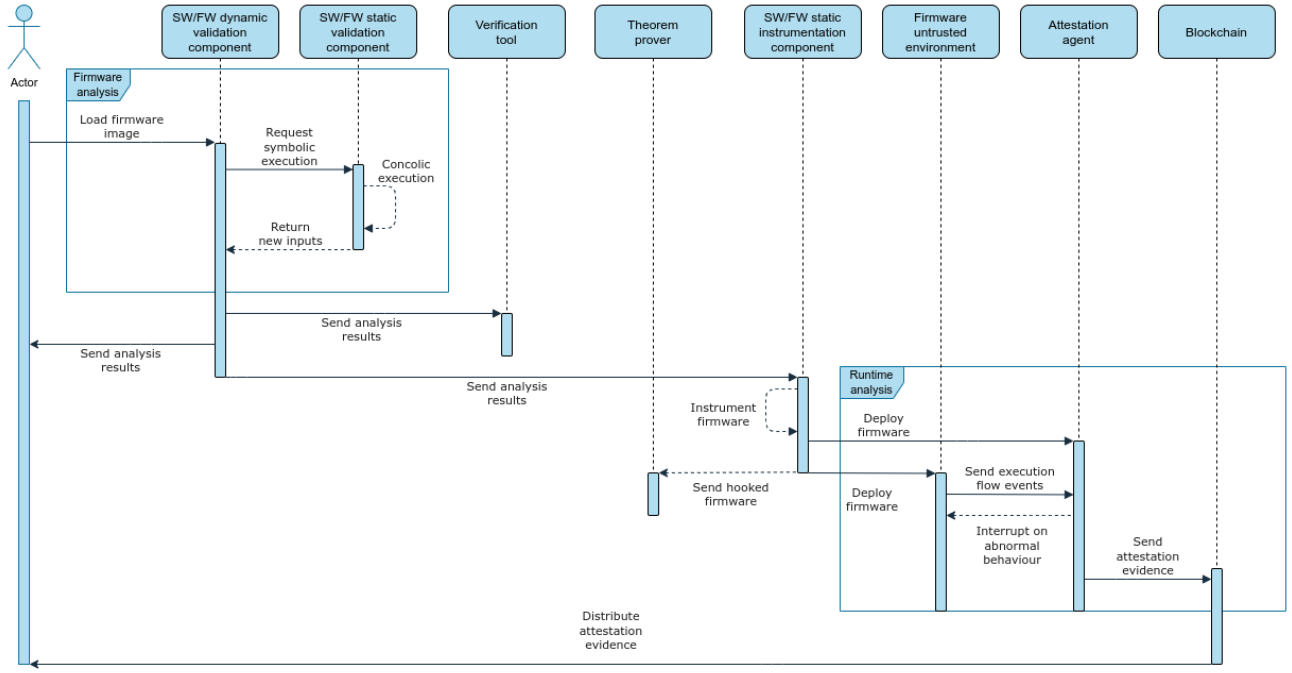

Figure 4.15: SW/FW Validation Workflow

4.4.7. Zero-touch onboarding

4.4.7.1. Description and functional objective

In terms of REWIRE, we desire secure and privacy-preserving seamless commissioning of devices into the network whilst providing device provenance and configuration integrity verification. The idea is for policy enforcement components to be able to instantiate the launch of the necessary enclaves as part of the underlying TEE, which is dictated by the system configuration.

A standardised approach for onboarding is the use of Manufacturer Usage Description (MUD) files which define the intended use of specific-purpose devices. MUD profiles are useful for security purposes to limit the attack surface of devices by defining policies and Access-Control Lists (ACLs) to restrict communication with other devices or services. Whilst the standard for MUD files gives a high-level method to automate deployment of IoT devices into a network in a secure manner, it does not specify the mechanisms for obtaining and enforcing such policies in a trusted and secure way. Trivially, a user could send their MUD file URL to the corresponding network components in the domain where it is to be deployed. However, this is insecure since a malicious user could forge its device identity and credentials to gain unauthorised access to the network. Thus, for secure and trustworthy MUD file obtainment during device bootstrapping, the Extensible Authentication Protocol (EAP) protocol is used in [REF-155] as part of the authentication, authorization, and accounting framework (AAA). The EAP-AAA protocol utilizes Pre-Shared Key (PSK) authentication, which is especially useful for lightweight devices which may be too resource constrained to handle public-key cryptography.

The design in REWIRE extends that of [REF-155] to use MUD profiles with the EAP-AAA merged with DAA. This approach of DAA leveraging ZKPs is needed to supplement EAP-AAA, since the latter protocol does not provide the privacy assurances necessary to REWIRE. The idea of using MUD profiles is to create the appropriate key restriction usage policies which need to be translated into enforceable rules. Such policies are used for protecting when each attribute-based signing key can be used, based on the output of the runtime attestation in which the mechanism for attestation is leveraging zero knowledge.

During the zero touch onboarding, to preserve privacy REWIRE requires the attestation of the satisfaction of policies in a zero-knowledge manner to access services, such that policies are hidden in the process.

That is, given information from the policy orchestrator a device should be able to attest to a MUD verifier that it is at an expected state, without releasing the actual state to the Verifier. If the output of this process is correct, then the MUD Verifier securely transmits these MUD policies onto the blockchain. Consequently, a secure and authenticated channel is established and used as a structure holding key restriction usage policies.

Note the following:

- The device must authenticate to the MUD Profile server to activate the long-term device identity key (which is pre-established) during runtime, as well as the SW update key. This identity key is then stored in the TEE.
- In REWIRE, threats must be identified and bound to the key restriction usage policies of the MUD profile.
- Attestation evidence is stored in the MUD file stored on the blockchain (BC). Moreover, a MUD file per device means that the user/device can update the attestation results to the BC during runtime.
- Attributes of the device, or a derived predicate of the attribute (Boolean assertion), may be included into the MUD profile itself.
- If attribute values are stored by the trusted component of Keystone, extra policies should be applied to ensure the trustworthiness of the device state. However, attributes do not necessarily need to be associated to key restriction usage policies.

4.4.7.2. Component workflows

Initially, keys need to be established between the device and Privacy CA to establish a secure and authenticated communication channel for onboarding the device into the REWIRE network. During this phase, the user of the edge device will request the MUD file from the Privacy CA (DAA Issuer), using a MUD URL related to specific components in the network. The MUD domain manager makes an authentication request, and the device responds with a list of attributes and a zero-knowledge proof (ZKP) of ownership. Note, proving ownership in zero-knowledge is important to preserve security. Following the device onboarding request, the Privacy CA interacts with the manufacturer domain, which includes interactions with the MUD server and AAA server. The former interaction is to request the device profile and policies, and the latter interaction is for the user and AAA server to mutually authenticate the user and the network. Then the identified policies are pushed to the REWIRE blockchain infrastructure so that to enable the trust-aware continuous authorisation and authentication processes of REWIRE and the attribute assess control mechanisms

The enrolment phase is needed in REWIRE to verify a device for access to resources in the network in a secure manner (authentication and authorization). To start, the user will request (using a MUD URL) the initialization of the enrolment phase. The privacy CA issues the MUD file to the MUD Verifier and issues DAA credentials corresponding to the URL sent by the user. The user proceeds to send the issued DAA credentials to the attestation agent (acting as the DAA prover) by creating a verifiable presentation leveraging ZKPs, such that the VP relates to attributes regarding the device status. Note, the policy orchestrator (PO), contained within the facility layer of the device, can be viewed as the MUD manager component in the MUD architecture of [REF-155]. The PO translates and maintains an updated version of the rules specified in the key restriction usage policies created during onboarding. The attestation agent accesses attestation policies from the PO, which are contained within the MUD file and a ZKP of the VP is sent by the attestation agent to the MUD (DAA) verifier to perform a verification check in zero-knowledge of attribute ownership. The device identity key, other public keys, and material from the MUD file are sent to the verifier by the secure enrolment agent to assist in the acceptance of device access to requested resources. Next, the verifier records and confirms the secure enrolment of the device to the BC wallet via secure oracles. Note that the MUD file and cryptographic keying material used in the enrolment process are stored in the protected TPM of the wallet in key hierarchies, such that, during runtime attestation the file and keys can be updated, revoked, and reviewed. The proceeding diagrams are split in two to demonstrate the zero-touch onboarding process, followed by secure enrolment.

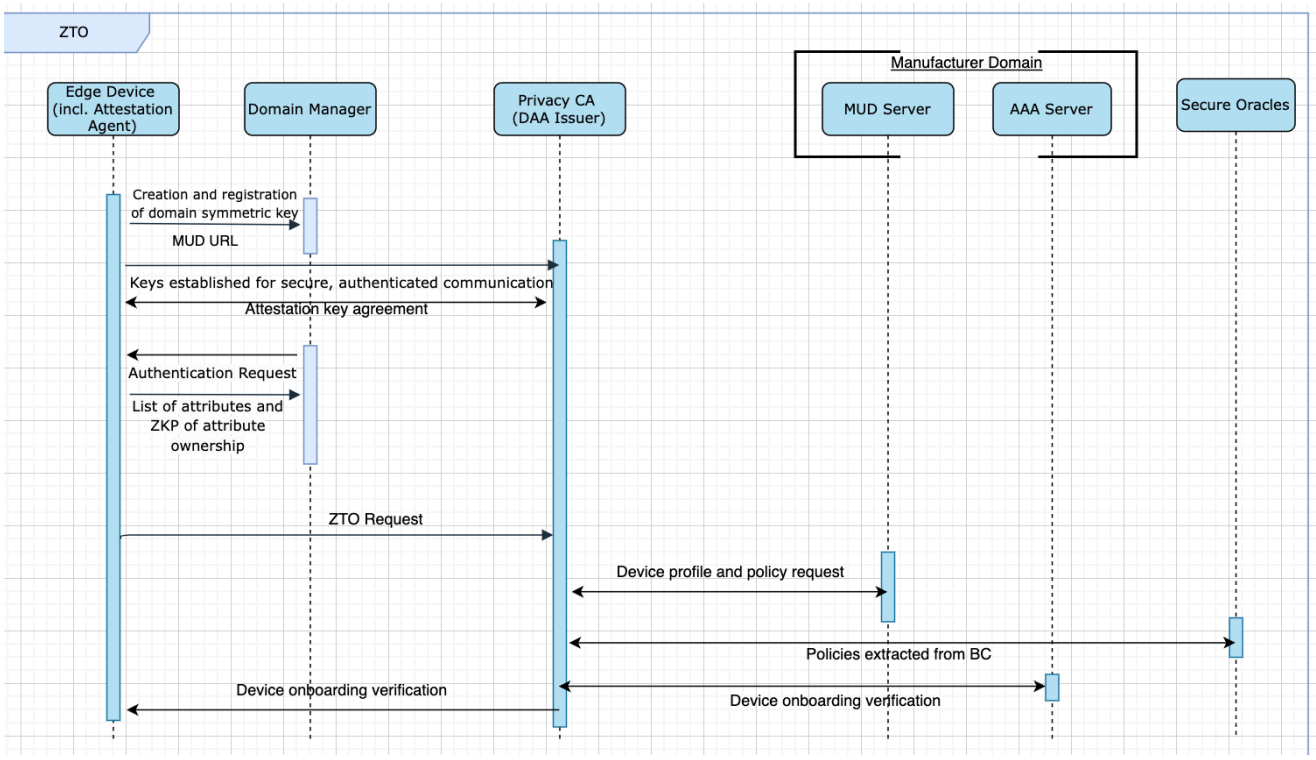

Figure 4.16: Zero-touch onboarding process

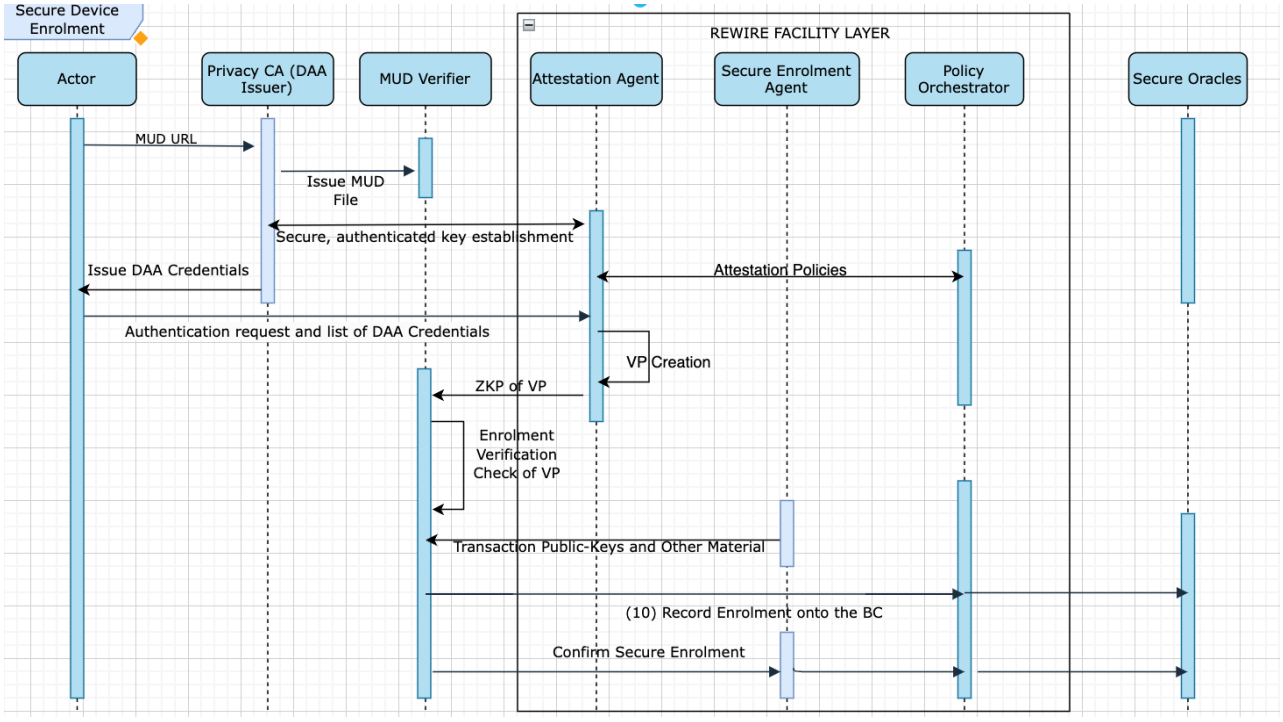

Figure 4.17: Secure enrolment

4.4.8. Software and Firmware Update Distribution Service

4.4.8.1. Description and functional objective

The IETF has emphasized the importance of simplifying the process of updating IoT devices [REF-156]. To address this, a SW/FW update service will be used to enable secure over-the-air delivery of patches for IoT devices. This component utilizes innovative distributed technologies to accomplish the task of

offering a comprehensive solution for managing devices, handling security patches, and coordinating their deployment. The system operates in a distributed manner, as the scale of some IoT deployments can be unmanageable with a centralized software distribution component. Instead, several trusted agents working from the cloud backed are synchronized leveraging on an inbuilt reverse proxy mechanism and the distributed ledger, enabling update rollouts of different multiplicities (one-to-one, one-to-many).

All events related to the Firmware/Software Update Over the Air are stored in the ledger, thus it contains critical evidence of updates along with end-devices profiles (such as status of the onboarding, current SW/FW version, addressing parameters etc.) needed by distribution agents. Furthermore, the SW/FW binaries are stored on the off-chain data storage providing a centralized repository that eases the synchronization of the distribution agents during the rollout process of an update. The transmission of update's binaries from a distribution agent to an end-device is supported by a secure channel established through an AE engine based on securely designed AES encryption mechanism and cryptographic materials that provide protections from side-channel attacks, authentication of updates and, in some cases, confidentiality of the update.

4.4.8.2. Component workflows

The diagram on Figure 4.18 shows the SW/FW packet management prior to its distribution. Through the previously mentioned reverse proxy mechanism, one of the distribution agents receives a vulnerability-free update package, after the SF/FW Validator component assesses the update package.

If the SW/FW package needs cryptographic protection, then it's sent to the REWIRE Crypto agility layer. The crypto layer can apply different cryptographic mechanisms to the SW/FW packet depending on the update multiplicity:

- One-to-one: the packet will be encrypted and signed using symmetric keying material.
- One-to-many. The packet will be only signed using asymmetric keying material.

It must be highlighted that the packet treatment by the REWIRE crypto agility layer is optional. If the packet does not need protection it can continue to the following steps of the SF/FW distribution without the crypto management. Whether the packet is protected or not it is submitted to storage through the Secure Oracle, which interacts with the BC infrastructure to record all the new changes. If the packet is successfully submitted its data pointer is returned to the distribution agent for future reference and to keep a SF/FW version control.

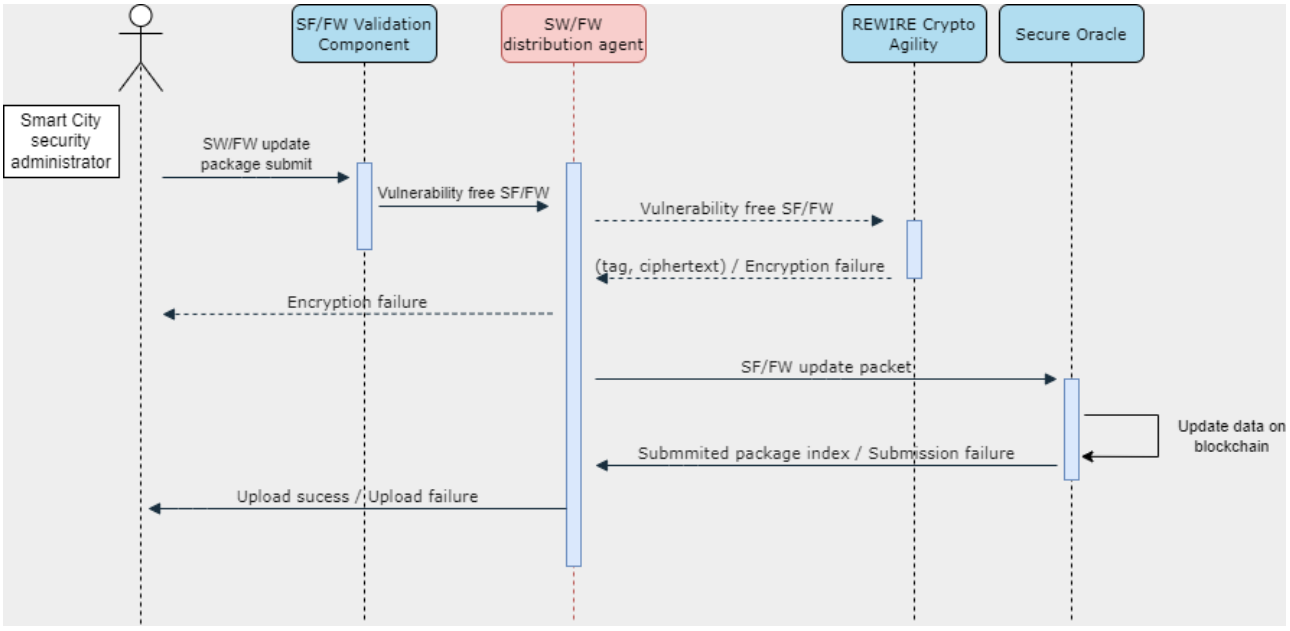

Figure 4.18: SW/FW Update Distribution

One-to-one update distribution

In Figure 4.19, the update distribution with a one-to-one multiplicity is showcased. To handle the update tasks to the distribution component it must first be verified that a privileged actor issued it. The update task will contain the indexes of both the target end-device and the update packet that will be installed, while the reverse proxy mechanism will handle this update task to the distribution agents. The distribution agent will recover the profiles of the to-be updated end-device through the ledger to confirm that the update is applicable. If the operation can carry on, the distribution agent will then launch an update query to the end-device through the Secure Oracle contained the SF/FW packet reference. Upon receiving the query, the end-device will send an ACK and request the Secure Oracle for the SF/FW update packet. This packet will be forwarded from the off-chain data storage to the end-device through the Secure Oracle. If the SF/FW packet is protected with encryption or signature the device will make use of the REWIRE SC-resistant AE to decrypt/check the legitimacy of the packet. Once this last step is finished, the device will obtain the SF/FW packet to install.

If both the decryption of the package and the update is successful, the end-device will send back an encrypted acknowledgment to the distribution agent. The distribution agent will receive and decrypt the acknowledgment while the Secure Oracle will update the BC data to reflect the new changes in the network.

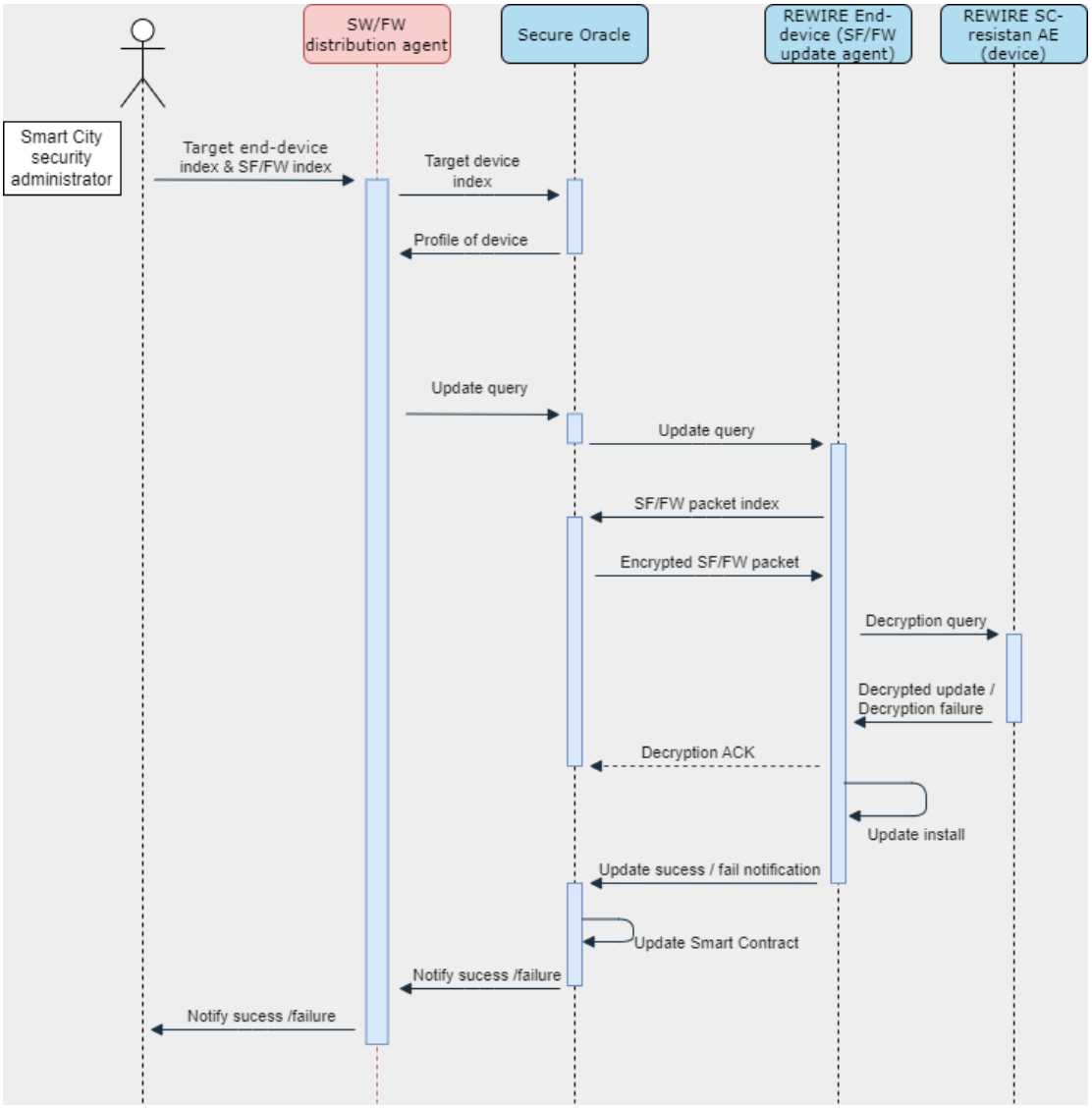

Figure 4.19: SW/FW Update Distribution Workflow (one-to-one)

One-to-many update distribution

Figure 4.20 depicts the update distribution with a multiplicity of one-to-many. The first deviation with the one-to-one multiplicity case is found when the update task is issued, since the update task will contain a list of end-device indexes which will be handled to one of the distribution agents through the reverse proxy mechanism. The agent will recover the end-devices' profiles from the BC infrastructure to check their information. The second deviation is the securitizing process of the update package as the multicast distribution of the update is not possible using symmetric key encryption. Instead, SW/FW packets are only signed using asymmetric keying material, which guarantees the authenticity of the packets, but no encryption is provided. In this case the packet distribution works just like the one-to-one multiplicity i.e., the end-device obtains the SW/FW packet from the BC infrastructure.

However, if encryption is needed an application-layer object-encryption [REF-157] protocol that is compatible with multicast connections can be used [REF-158]. The distribution agent and end-devices will first perform a lightweight authenticated key exchange [REF-159] to generate the asymmetric cryptographic material used by the encryption protocol. In this case the SF/FW will not be obtained from the off-chain data storage, instead the distribution agent must launch the update query alongside the SF/FW packet itself, which will be handled to the end-device through Secure Oracles.

Although the encryption-decryption of SF/FW is done using the object-encryption, the acknowledgments of the signature check and correct decryption are securitized with the symmetric AE just like one-to-one update case.

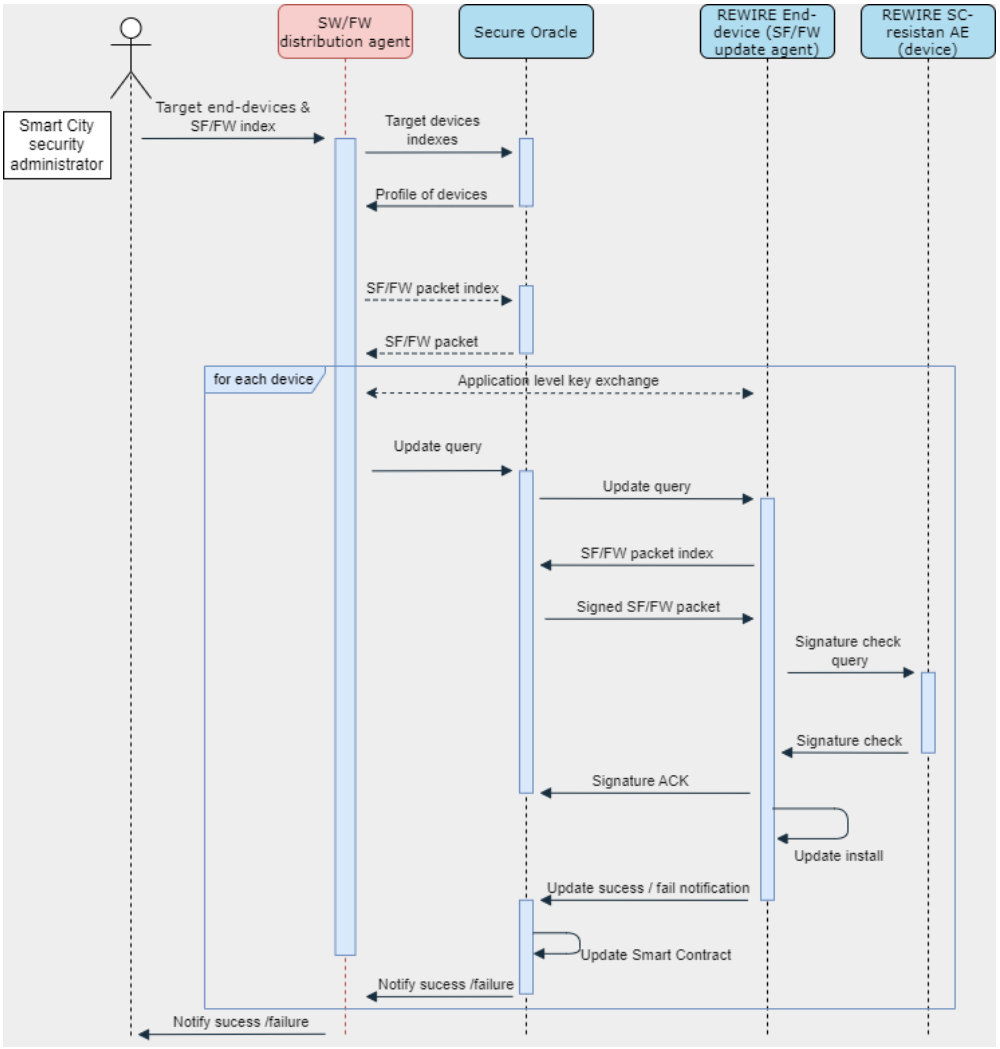

Figure 4.20: SW/FW Update Distribution Workflow (one-to-many)

4.5 REWIRE-enabled Edge Device and Services

This section, as its name suggest, is focused on the REWIRE-enabled edge devices and the supported services. The REWIRE-enabled device (recall the conceptual architecture in Figure 4.1) contains the REWIRE Facility Layer, as an intermediate layer for interaction between the agents and the REWIRE backend infrastructure.

4.5.1. REWIRE Facility Layer

This layer is part of each REWIRE-enabled device that facilitates all the necessary agents of REWIRE. In other words, is the intermediate entity between the device itself and the backend infrastructure, providing the necessary interfaces to trigger and interact with the agents (e.g., Attestation Agent). It also contains the Policy Orchestrator as the main component to orchestrate and enforce the received policies to the agents. More details are provided in section 4.5.6 below.

4.5.2. Device trust-aware secure enrolment

Secure onboarding of devices into the REWIRE network needs to cover trust-aware enrolment to better capture the “Trust vs. Privacy” trade-off we will consider when designing the ZTO protocol. In Section 7 there is a discussion on zero trust security modelling which requires strict identity verification for every subject/entity onboarding. REWIRE requirements and functional components are designed to satisfy the pillars of a Zero-Trust Architecture (ZTA) in the following ways.

Enhanced identity governance is an approach to develop a ZTA [REF-160] using the identity of actors as the core component of policy creations. The core idea of our ZTO protocol (see Section 4.4.7) is to link a device to its MUD profile and key restriction usage policies. Adopting a ZTA implies that devices are managing their own credentials, in turn, implying the design of an SSI [REF-161] management system to maintain privacy preservation. Thus, devices can selectively disclose attestation evidence in the form of verifiable presentations (VPs), sent to the authority (MUD Verifier) checking the operational assurance of the device. That is, to check integrity and certify the profile of a device. Note, runtime attestation evidence is formed by the continuous monitoring and measurements (behavioural and environmental assets) taken by REWIRE components.

Verifiable credentials (VCs) can enable holders to prove their attribute-based credentials and they are a component of verifiable presentations (VPs) which are used for authorisation, whilst considering device trustworthiness and privacy considerations. Credential properties that should be considered for device onboarding in REWIRE includes, refreshing (updates), delegation, and blinding attributes. More importantly, the selective disclosure of attributes is important in REWIRE to preserve the privacy of specific attributes. Observe, the authors of [REF-162] provide a non-technical but useful summary of Anonymous Credentials and how to categorise their definition of ACs, based on how they achieve their core properties and how they are used. Moreover, the authors define two separate strains in which credentials are operated using zero-knowledge proofs OR re-randomised (blinded) by the prover between showings. Note, group signatures can be used as an alternative tool to ACs.

In the ensuing, we focus on Attribute-based Signature (ABS) schemes as we believe ABS compliments the properties and requirements of REWIRE’s architecture. ABS schemes allow users to obtain attributes from issuing authorities, and sign messages whilst simultaneously proving compliance of their attributes with a verification policy. ABS demands that both the signer and the set of attributes used to satisfy a policy remain hidden to the verifier. Moreover, Hierarchical ABS supporting roots of trust and delegation was recently proposed in [REF-163] to mitigate problems in the centralised setting of ABS, such as scalability. Therefore, HABS is even more useful to the decentralised setting we propose in REWIRE.

The Hierarchical Attribute-based Signature (HABS) primitive was introduced in [REF-163] to support delegation of attributes along paths from the top-level authority down to the users while also ensuring that signatures produced by these users do not leak their delegation paths, thus extending the original privacy guarantees of ABS schemes. The generic HABS construction also ensures unforgeability of signatures in

the presence of collusion attacks and allows for a dedicated tracing authority to identify the signer and reveal its attribute delegation paths such that the scheme has public verification to hold the tracing authority accountable.

Furthermore, device onboarding in REWIRE must consider the converse process of revocation. In [REF-164], revocation of credentials (performed by the issuer,) should not reveal any identifying information about the subject, the holder, the specific verifiable credential, or the verifier. Moreover, the issuer can disclose the revocation reason. Lastly, Issuers revoking VCs should distinguish between revocation for cryptographic integrity (for instance, corruption of the signing key) versus revocation for a status change.

With respect to ABS (and HABS), revocation can be applied to signers or some of the attributes they possess, and it is a challenging property to ensure but equally important for privacy-preservation. Alas, the authors of [REF-165] highlighted that ABS schemes in the literature lacked efficient revocation of either signers or their attributes, relying on generic costly proofs. Resolving this problem was the main motivation for proposing the new HABS primitive in [REF-165]. Unfortunately, in HABS there is a further need to support revocation of authorities on the delegation paths, which is not provided by existing HABS constructions. To counter this issue, [REF-165] proposes a novel Verifier-Local Revocation (VLR) property which extends the original HABS security model of [REF-163] to address revocation and develop a new attribute delegation technique with appropriate VLR mechanism for HABS. Moreover, their scheme is the first to be based on lattices, thought to offer post-quantum security. It may be of interest to explore post-quantum security of attribute-based VPs in the context of REWIRE.

Looking at the current ways to implement such schemes, we highlight document [REF-166] which is a draft with respect to linked data cryptographic signature suite registries. The contributors summarise several suites and their corresponding verification methods currently known to the community. Specification [REF-166] was published by the credential community group, however, it is important to note that the specification is not yet a standard regarding W3C. Of particular interest is the BBS+ Signature 2020 Suite. Recall, BBS digital signatures schemes are utilised for short group signatures supporting beneficial properties such as evaluating multiple messages to producing a single output digital signature. This desirable property enables the possessor of a signature to generate proofs that selectively disclose subsets of the originally signed set of messages, whilst preserving the verifiable authenticity and integrity of the messages. Importantly, these proofs are said to be zero-knowledge as they do not reveal the underlying signature; instead, what they reveal is a proof of knowledge of the undisclosed signature.

The aforementioned methodology will be followed in order to create a ZTO protocol in the context of REWIRE. The main entity on the edge device that is going to implement this logic in the ZTO agent, as part of the REWIRE facility layer.

4.5.3. Device state management

4.5.3.1. Description and functional objective

Enclave migration based on RISC-V is one of the innovations proposed for REWIRE. Enclave migration can be useful in a number of scenarios: maintenance of a device, risk situation or as a result from a previous attestation. It enables the system to securely migrate and send the state of an enclave running in an IoT device, to a remote IoT device, which will initialize a new enclave running the original application and restoring the original state.

On a theoretical level, enclave migration can take place inside the same machine (machine A, enclaves A_1 and A_2), or between two machines (A and B) and enclaves (enclave A, enclave B). This latter case is based around the concept of state storage and remote initialization of an enclave such that it contains the original eapp (not sent) plus the original state. This chapter presents the second case since the first one is a simpler subset of it. In this case, the state of enclave A in machine A is securely migrated to a different enclave B in machine B, without service interruption.

4.5.3.2. Component Workflows

Two main steps must sequentially happen, with some conditions:

1. Establishment of a secure channel
    - After a certain component (e.g., the Attestation Agent of TEE A) triggers the enclave migration between TEE A and TEE B, the system must establish an ephemeral secure channel between the SMs of both machines. This is not available by default.
    - Assuming the SMs have a private key (delivered from the Root of Trust), and that they know the Public Key of the other SM, and that Public Key is certified by a trusted authority, they can agree on a private key using a Diffie-Hellman algorithm and establish that ephemeral secure channel.
        - A different approach would be for the SMs to have a private DAA, not necessarily the same as the private key delivered by the Root of Trust, and that is certified by a CA; and use this private DAA to authenticate them both during this process.
    - The Diffie-Hellman code can be run inside the Keystone SM
2. State storage and migration
    - Once there is confirmation that the secure channel is established between SM A and SM B, enclave A encrypts the state with a key which will be named K_{MI} (derived from the private SM A key) and stores that state. State may go to persistent memory.
    - After that, the state can be sent together with the encryption key using the secure channel, and then the other machine should be able to decrypt and restore the state.
    - When this is done, the application in the second machine can be run in enclave B, using the state/data that has been restored from enclave A. Note that the application itself ($eapp_A$) is not sent: other actor must inform machine B that $eapp_A$ is the eapp to run in enclave B.

It is important to note that some intelligence must actuate outside the boundaries of the SMs and enclaves involved in the process. This is the untrusted world must activate enclave migration if triggered by conditions that are external to the SMs and enclaves.

In the description of this feature, the following elements are taken into account:

- Policy Manager: this is an abstraction for any entity in the REWIRE cloud backend able to, as said, determine that enclave A must be migrated to enclave B. Not active in runtime and thus not included in the sequence diagram.
- Attestation Agent A is, in this case, the component that triggers enclave migration. However, other components could in principle trigger the migration process (in the case of e.g., maintenance). It belongs in the untrusted world.
- Device State Management Agent is the component that internally (to each TEE) orchestrates the necessary operations for enclave migration.
- Enclave A, Security Monitor A, Enclave B and Security Monitor B are self-explanatory.

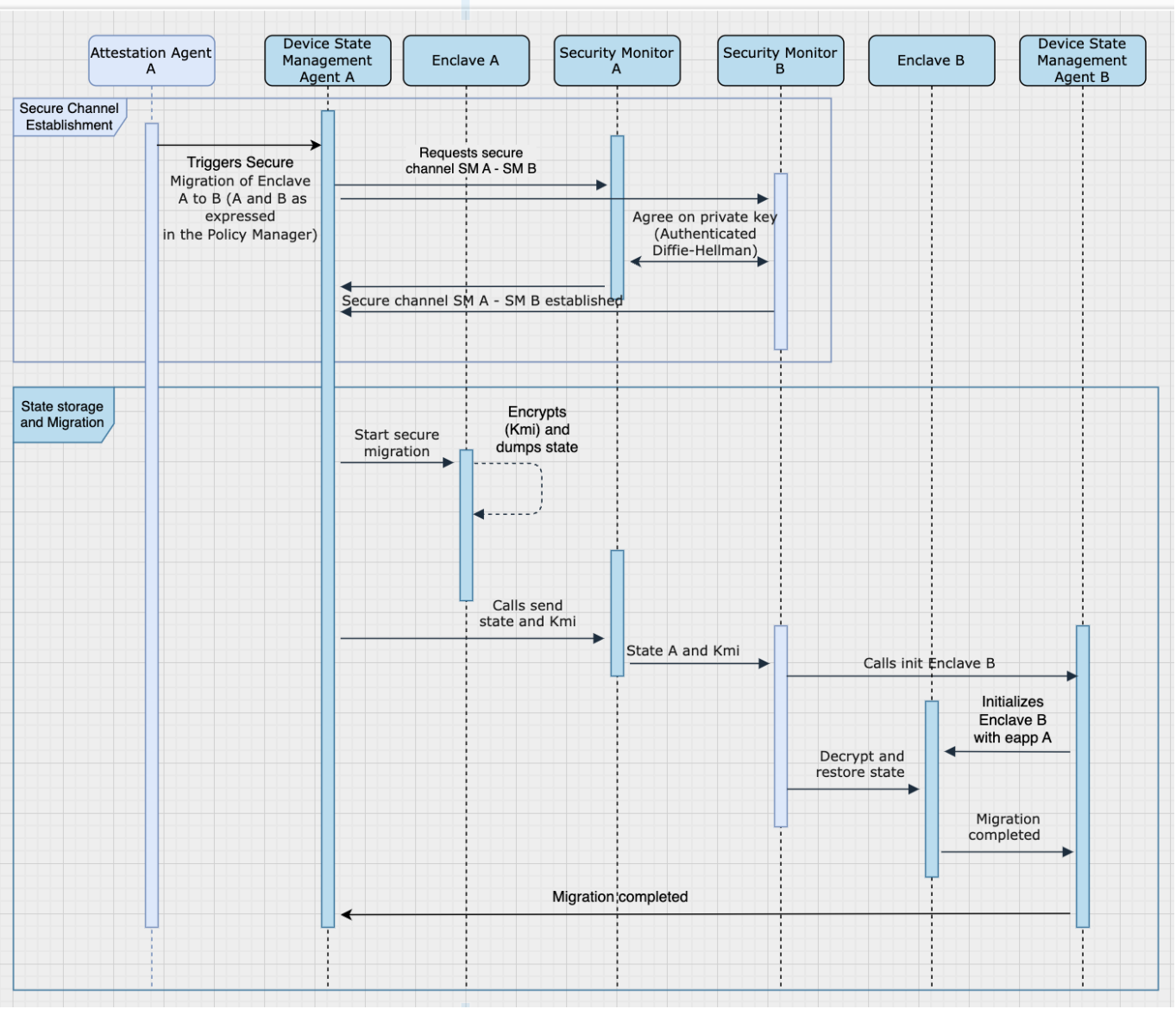

Figure 4.21: Device state management workflow

4.5.4. Secure Upgrade

Secure Upgrade based on RISC-V is another of the novel functions proposed for the REWIRE TEE. This function is one of the core elements of the full SW/FW update feature of the REWIRE system. As a core element of it, it requires that several pre-requisites are fulfilled in advance:

- The Secure device on-boarding has been successful, and a master key has been received by the Security Monitor from the device's RoT.
- The Software and Firmware Update Distribution has been successful, and the software update has been decrypted.
- The Firmware & Software Validation has been successful.
- The software update must be converted to an eapp.
- The software update is already available in (local) memory
- It is assumed that a single enclave (Enclave A) is running the old SW
- The Security Monitor, through the Policy Manager (design time), knows which enclave (Enclave A) is running the old software.

4.5.4.1. Description and functional objective

The primary objective of Secure Upgrade is securely updating the old software that is being currently run for a new, superior version of the software, without loss of state data.

4.5.4.2. Component Workflows

Secure Upgrade shall take place inside the same machine (machine A, enclaves A and B). It revolves around the concepts of (1) (original) enclave state storage and (2) initialization of a new enclave such that it contains the new eapp plus the original state.

Sequentially, it follows the next steps:

- The SW/FW Update Agent instructs the Security Monitor to start the Secure Upgrade procedure.
- The Security Monitor (SM) verifies one or more characteristics of the new software (e.g., versioning) against the State Dynamic Map. Note that this has not been included in the sequence diagram for the sake of simplicity. This could also involve checking an external whitelist database of software versions.
- The SM instructs Enclave A to encrypt and seal (store) its state with key Ksu.
- Ksu is a derivation of the master private key (Master Secret) and it is created on demand. It is not stored. It could be computed as H(MS, ID, context) where H is a hash function, MS is the Master Secret, ID is the public key of the enclave developer, and context would just be a string like “upgrading” – as used to derive different keys for different contexts.
- Overall, we need a sealing key that should be available to both the old and the new enclaves. Hence all enclaves that are signed by the same developer and carry the same developer public key could have access to this sealing key. Having two enclaves that can access the same sealing key, allows to transfer state via the sealing interface. That is, one enclave seals state and the other one unseals it. This is similar to the Intel SGX procedure when sealing keys are obtained with the MRSIGNER option.
- Enclave A stops all threads, encrypts and seals its state. Enclave A then calls back to Security Monitor.
- The SM destroys Enclave A. Memory is freed. Then updates the State Dynamic Map with an in-progress status and calls back to the SW/FW Agent.
- The SW/FW Agent allocates the new EPM, and initializes it with the enclave’s page table (PT), the runtime (RT), and the eapp for the new SW update – then instructs the SM to protect that Enclave B.
- The SM protects Enclave B and instructs it to use Ksu to decrypt and restore the original state.
- Enclave B decrypts and restores state, then calls back to the SM.
- The SM, after verifying and measuring the new Enclave B, updates the State Dynamic Map and calls back to the SW/FW Agent with a success value.

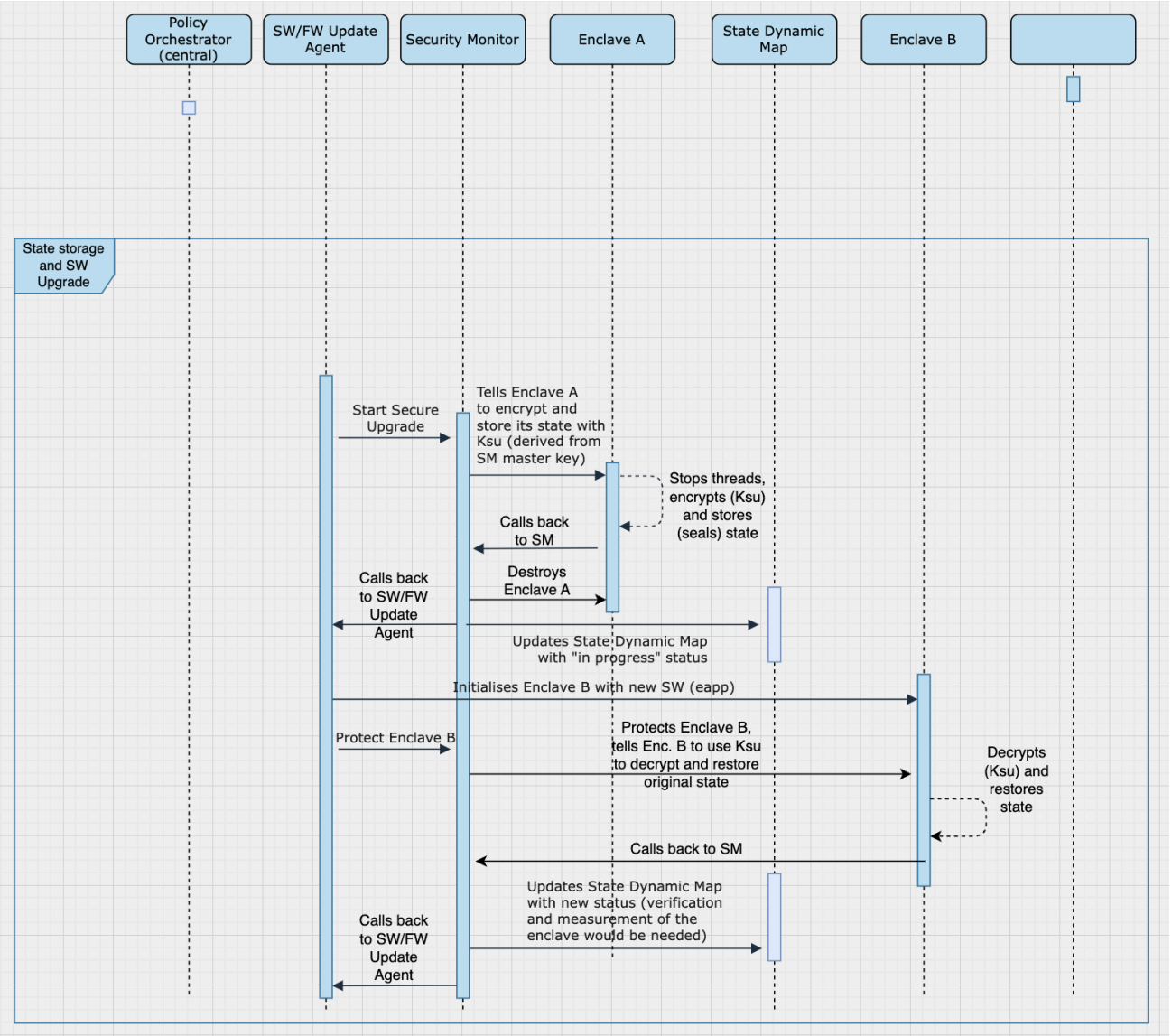

Figure 4.22: Secure Upgrade workflow

4.5.5. Attestation Agent

4.5.5.1. Description and functional objective

Operational assurance of IoT devices is of paramount importance in order to ensure their trustworthy operation in the context of collaborative infrastructures. Attestation is a security service in order to validate the integrity of an entity such as an IoT device or an installed software. Static attestation mechanisms enable the detection of manipulation of a device's static memory content (e.g., the program code or program's configurations). Runtime attestation mechanisms provide information about a device's runtime behaviour, allowing the verifier to detect also dynamic attacks (e.g., code-reuse attacks). REWIRE utilizes runtime attestation for verification of edge devices' operational assurance.

4.5.5.2. Component Workflows

In particular, REWIRE offers **efficient attestation mechanisms** covering all phases of a device's execution; from the trusted boot and integrity measurement of an IoT device to the runtime behavioural attestation of safety critical components of a system providing strong guarantees on the correctness of the control- and information-flow properties, which will directly supported by the underline trusted

component (e.g., the REWIRE's TEE).

REWIRE's layered attestation allows attestation to build the global picture of a system's integrity from the bottom up, one layer at a time. Such an approach enables REWIRE to provide a more nuanced view of the device state as it can isolate integrity and operational correctness violations identifying exactly which portions of the system are trusted or not. The REWIRE Attestation engages a gamut of cryptographic operations which rely on the secure management of cryptographic keys. Thus, it becomes clear that we need to adopt the best practices in order to manage this wide variety of keys in a secure manner. One core feature of the REWIRE Attestation mechanism is the **adoption of key restriction usage policies** in order to bind the usage of the key to a device state that is known to be correct and trustworthy, ensuring that the REWIRE cryptographic enablers are not used by a device that has been compromised by a malicious party. In parallel, the adoption of key restriction usage policies minimizes the attack surface on the target device. The following sequence diagram illustrated the interactions that will take place in the context of the attestation agent.

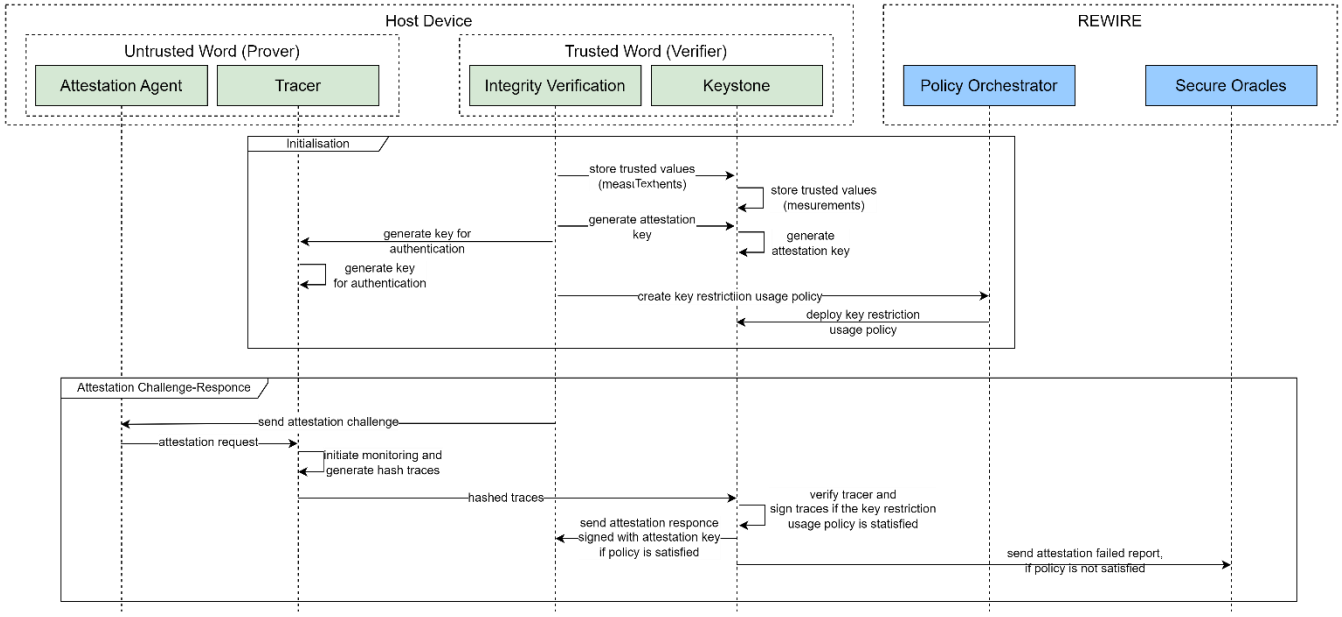

Figure 4.23: REWIRE Attestation Agent

4.5.6. Policy Enforcement

Policy Enforcement is a core functionality of the REWIRE device, as part of the Policy Orchestrator. More specifically, the Policy Orchestrator is the entry point that receives enforceable policies (generated during the design time phase or the security administrator during runtime) and decides in which agent will be enforced/applied. Thus, the Policy Orchestrator is responsible to run the necessary piece of logic for enabling and retrieving the necessary information by the agents. In other words, the functionality of Policy Orchestrator is twofold: a) it manages and orchestrates the received policies for triggering the appropriate agent and b) facilitates the communication between the agents and the backend infrastructure through the Secure Oracles.

4.5.7. Key management

REWIRE will offer a key management mechanism in order to manage the cryptographic keys that will be necessary for supporting the secure and trustworthy execution of all the envisioned functionalities of REWIRE. For instance, the SW/FW update process, the enclave migration for the device state management, the ZTO, are some examples of services that capitalise on the cryptographic keys. Hence, REWIRE will offer an holistic key management mechanisms for the generations and protection of

cryptographic keys. Different keys will be used in order to support the various crypto-based services. The key management mechanism will be using a key hierarchy where keys will be created to serve different purposes. The key management mechanism will be an integral component of the security monitor of the TEE. In D2.1 we elaborate on our initial efforts on defining the type of keys that the REWIRE services are going to need, but this work will be completed in the context of WP3 and WP4.

The hardware based long-term device ID key and the authenticated encryption (AE) symmetric key (for the SW/FW update) are created in the device setup and distributed to the KMS in the security monitor (SM). Note, the AE symmetric key is onboarded on the device during the manufacturing process as explained in Section 4.3.2. The asymmetric attestation key pair is generated from the HW RoT, using randomness, and is sent to the SM and ZTO agent. The SM key-pair is bound to the SM identity and trusted HW component root key. This is essential for processes relevant to the key hierarchy, such as *data sealing,* not depicted on the diagram. Data sealing allows an enclave to derive (using KDF) a key for encryption to be able to save data in untrusted, non-volatile memory outside of the enclave. The SM has a key management component whereby the master/root keys of key hierarchies are used to establish children keys (encryption, signing etc.) using a key derivation function (See Chapter 8). The ability to generate keys in this manner is fundamental to the numerous cryptographic protocols and operations in the REWIRE network, such as runtime attestation, to bind the usage of the key to a device state in line with the key restriction usage policies created in the ZTO process.

In the context of the ZTO process, the device ID key is sent to the agent, and onboarding proceeds (see Section 4.4.7). Once the device has been authenticated and onboarded, the SW Update key is activated and the DAA Key is created. Note, a separate key hierarchy is created for the SW update key to provide segmentation from the key hierarchy of the DAA key and attribute-based signature keys. During the onboarding phase, the user seeks to authenticate the device to the MUD verifier in the REWIRE network. This is achieved by the device creating Verifiable Presentations (VPs) of credentials without disclosing the device identity, using zero-knowledge attribute-based signatures or attribute-based DAA. In either case, attributes will be depicted as keys in an access tree structure, generating a matrix of attributes, based on which attribute-based signatures (ABS) are created. Attributes in a verifiable credential (VC) can represent a row in the matrix, and a predicate/operation over these attribute keys, or a subset of attribute keys in the row, can be viewed as an accumulator producing a signature. Note, the attribute keys do not necessarily need protecting under a key restriction usage policy, however, the device DAA key does need protection since it will be used for anonymously signing the ABS's to create a blinded VP. For the device to be able to use the DAA Key, it will have to check the correctness of the key restriction usage policy set for using it. The policy orchestrator feeds the VPs with security claims, and the key restriction usage policies, to the secure oracles. During runtime, the user can selectively disclose their attributes by utilizing decentralized ABAC which REWIRE supports. Specifically, the collected data gets fed to the blockchain infrastructure for off-chain storage in a digital wallet supporting the SSI paradigm, components of which are DIDs and VCs. Cryptographic keying material is also protected in the BC wallet in a KMS.

The abovementioned key management and key derivation approaches reflect the current state of the developments of REWIRE. We need to highlight that these approaches may be revised in the context of the technical deliverables of the projects, as new challenges will arise during our developments.

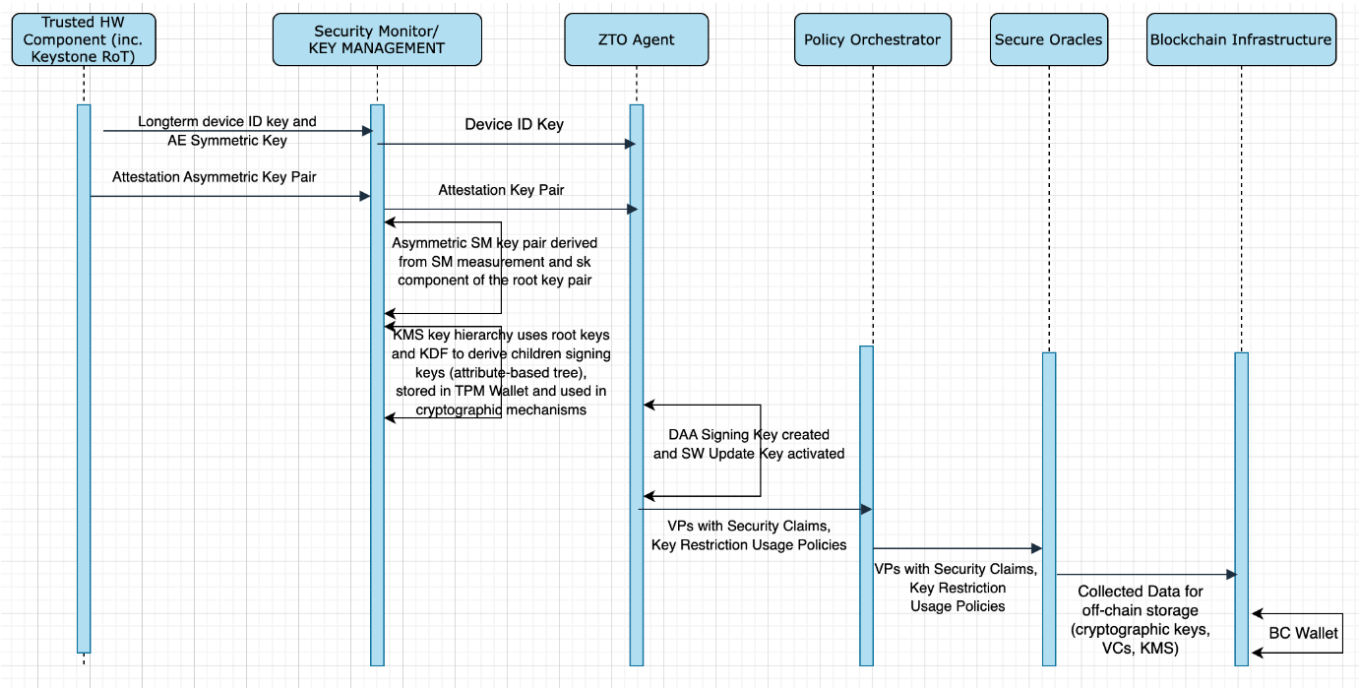

Figure 4.24: REWIRE Key Management

Chapter 5
REWIRE Framework Requirements

This is a core chapter of deliverable D2.1 since it focuses on the REWIRE framework requirements. More precisely, the requirements are divided to functional, security, operational assurance, formal verification, and RoT requirements. The requirements were gathered on the methodology described in Chapter 3.

5.1 Functional Specifications and Requirements

ID	FR.FR.1 (Mandatory)
Title	**Dynamic awareness of potential vulnerabilities and threats and complete overview of the deployed environment**
Actors/Components Involved	Security Administrator, Risk Assessment, AI-based Threat Intelligence, SW/FW Validation Component, Attestation mechanisms (Attestation Agent, REWIRE Tracer, Integrity Verification), Secure Oracles, Blockchain infrastructure.
Description	**Background:** The REWIRE framework should be able to collect and report information regarding vulnerable assets/devices or assets being under attack. In order to be able to assess the risks and the security posture of a highly distributed environment with several interconnected heterogenous devices, the various underlying threats and vulnerabilities, there is a need to identify risks so that to proceed with the enforcement of new operational and security policies for regulating the operation of the environment or even to proceed to updates on the HW and SW co-design on systems (following the design-time phase of REWIRE). **Description:** REWIRE needs to provide to the administrator the necessary tools to perform dynamic and (semi-)automated risk assessment. To do so, REWIRE needs to take advantage of proper visualization technologies which can offer a graphical representation of the monitored environment and ease the administrator to keep track of the existing interconnections among the numerous and heterogenous devices. When new vulnerabilities or threats are detected by the REWIRE artifacts (e.g., AI-based Threat Intelligence) or reported by the community for known assets, this graphical representation will assist the administrator to evaluate how the interdependencies of the assets could trigger cascading effects and increase the cyber risk for critical services. REWIRE needs to provide solutions which are aligned with the good practices and standards to enable the dynamic risk assessment of IoT environments and safety critical applications. That is, based on evidence, the risk assessment tool will perform an informed analysis over the IoT deployment being monitored, and will inform the system administrator about its security posture. This functional requirement could enhance the runtime monitoring capabilities of REWIRE and assist the administrator to take informed decisions for mitigation actions or attestation policies. The outcome of the risk assessment process is a risk report which will provide an overview to the administrator in order to take informed decisions and trigger actions. **Remarks:** – Risk assessment needs to provide an output and highlight the identified risks and feed the HW/SW co-design formal verification processes. – The term dynamicity refers to the ability of the framework to acquire and process events from a monitored topology in near real-time manner.
Technology enablers	Common Vulnerability Scoring System (CVSS), Common Attack Pattern Enumeration and Classification (CAPEC), MITRE ATT&CK, Graph databases (e.g., Neo4j)
Connected To Other	• **FR.FR.16**: **AI-assisted misbehaviour detection:** The risk assessment

Requirements	methodology and tool of REWIRE needs to take inputs on the detection of security or misbehaviour incidents.
Impactful Attacks and Mitigation Measures	All attack vectors should be able to be represented in the risk assessment environment of REWIRE. Given the risk assessment output, the administrator should be in position to take informed decision for mitigation actions.
KPIs	• Elapsed time to complete a risk assessment execution for different specific scenarios. • Scalability of the risk assessment engine w.r.t the number of assets/vulnerabilities/threats engaged.

ID	FR.FR.2 (Mandatory)
Title	**Secure remote asset management and reconfiguration effectiveness**
Actors/Components Involved	Security Administrator, Blockchain infrastructure, SW/FW Validation Component, SW/FW Distribution Service, Secure Oracles, SW/FW update (on the device), SC-Resistant AE
Description	**Background:** The capability to remotely update IoT devices is a critical aspect of ensuring the long-term security of these assets. Devices should be able to receive SW/FW updates and configuration data in a cryptographically secure manner, utilising crypto protocols which are resistant against attacks. End-devices in Smart Cities scenarios are deployed in vast geographical areas and operate without human supervision. Furthermore, they even lack keypads or displays. For these reasons, to ensure the right operation of the network, remote management and reconfiguration are essential. Not only these must be implemented, but also, they require a high level of security since these devices might manage critical assets — fire detection and suppression systems in public areas, access control to public premises, waste management, plague control, street lightning, traffic control. Currently most automotive SW updates are performed in an authorized dealership for SW updates regarding both new functionalities and bug fixes. At the same time, new automotive technologies in the field of autonomous driving and V2X connectivity require constant SW updates throughout the product lifecycle as well as day-to-day operation. Therefore, it emerges as a necessity to remotely perform SW updates (OTA) and this needs to happen in a secure manner as it can affect road and traffic safety. The same requirement for secure remote asset management and reconfiguration effectiveness applies also for the Spacecrafts applications and services. In fact, updating the SW/FW of a satellite which is in orbit around Earth is a rather challenging task, as the update procedure takes place in segments, utilising multiple ground stations communicating with satellites for only limited time frames. Secure and reliable SW/FW update processes are crucial of the domain. **Description:** REWIRE must allow the Security Administrator (e.g., network owner, manufacturer / supplier) of the required tools to perform remote distribution of FW or SW data — i.e., selecting a FW binary or SW configuration files and the target devices from a list that must receive this information in a timely manner. Additionally, FW or SW data must be validated before the deployment. There is a risk of attackers triggering rogue software updates — e.g., malware or tampered FW or even legitimate developers may unintentionally create vulnerable SW/FW binaries. To do so, REWIRE will ensure security during SW/FW remote transmission and validation upon reception capitalizing on openly standardized protocols and data formats to guarantee the seamless integration of several types of devices within the domain. In Smart City scenarios, due to their heterogeneous nature, target devices will implement resource constrained MCUs that encourage employing lightweight

	security and communication protocols and mechanisms. In contrast, for vehicular scenarios, due to the typically high processing power of ADAS ECUs, more computational resources will be available, and thus, stronger security protocols can be considered. This can significantly improve the secure transmission of SW packages to the targeted vehicle(s), preserving security and integrity of the SW update package. A generalized SW update or bug fix can be globally deployed on control units of all cars of a given brand or model, simplifying the update procedure, and reducing the attack surface. Therefore, REWIRE can provide the OEM / Authorized supplier with the ability to select a specific SW package and automatically roll it out to specific vehicles or the whole OEM fleet (in case of a general update). Hence, remote SW/FW update will facilitate proper operation of autonomous driving and V2X connectivity (e.g., ensure that the vehicle is always up to date regarding HD road maps). In the context of smart satellites, secure SW and FW update processes will be developed in order to ensure that not only the SW/FW update can take place utilising multiple ground stations, but also the update operation will be fail-safe and will ensure that, even in the case of an update fault, the satellite will remain operational as rollback mechanisms will ensure that the system will always be in an operational condition. This SW/FW must be carried out from the source to the destination privately and without tampers through the whole infrastructure. In addition, in the case of an already successful attack, a SW/FW update will offer a chance to regain full or partial control of the already compromised, safety relevant or not, components of the vehicle's system. Moreover, OTA updates can immediately deploy SW enhancements or new functionalities (e.g., improved perception algorithms with edge-case training), allowing continuous improvement throughout the product lifecycle. However, due to the nature of massive deployments, this operation should also enable the transmission of the SW/FW configuration data to several devices concurrently. Since this is not scalable in terms of time, computational resources or radio bandwidth, multi-cast secure communication technologies and methods should be available instead. Finally, after a successful or failed software update procedure takes place, this should be reported back to the administrator through the secure remote asset management tools. This information should be confidential. **Remarks:** – The tools must receive FW or SW configuration data package from an authorized user as input, which will be in turn transmitted as an output through the infrastructure to the devices/vehicles/satellites. – The SW/FW package should meet the REWIRE framework security standards. – Despite the "one-to-one" nature of automotive SW/FW updates that will be considered on REWIRE, one-to-many case can optionally also be taken into consideration, for the case of Smart Cities scenarios or updating entire (or large groups of) vehicle OEM fleets. Therefore, both one-to-one and one-to-many scenarios can fall under security clearance. – Devices with heterogeneous computational or bandwidth requirements must be supported — openly standardized protocols and data formats. – A report of the success or failure of the procedure must be presented to the administrator through the remote asset management tools. This information should be confidential. – User should be informed whether the SW/FW update has been successfully deployed. This information should be confidential.
Technology enablers	Hyperledger Fabric, Keystone TEE
Connected To Other Requirements	• **FR.FR.5**: **SW/FW unpacking and vulnerability analysis:** The SW/FW binaries need to be validated and be sanitised from known vulnerabilities before the SW/FW update process takes place. • **FR.FR.1**: **Dynamic awareness on potential vulnerabilities and threats**

	and complete overview of the deployed environment: The Risk Assessment needs to consider potential identified vulnerabilities and threats.
Impactful Attacks and Mitigation Measures	SW/FW updates and configurations should be performed in a cryptographically secure manner utilising crypto protocols which are resistant to side-channel for recovering authentication keys. This requirement itself is necessary as a mitigation measure.
KPIs	• Computational load for security mechanic, mean time for update completion, overall network requirements, time for mitigation deployment • Timeliness of SW/FW Distribution: Measure the time it takes for the SW/FW data to be successfully transmitted and received by the target devices or vehicles. This KPI ensures that updates are delivered in a timely manner to ensure the continuous operation of the network. • Validation Success Rate: Measure the percentage of successfully validated SW/FW data upon reception. This KPI assesses the effectiveness of the validation process, ensuring that only trusted and authentic updates are deployed, mitigating the risk of malware or tampered firmware. • Security Protocol Overhead: Measure the impact of security protocols on the communication overhead. This KPI assesses the efficiency of lightweight security protocols in resource-constrained devices in terms of bandwidth employed. • Efficiency and Success Rate: Measure the efficiency of multicast technologies in transmitting SW/FW data to multiple devices concurrently. In addition, the success rate of one-to-many SW/FW updates, especially in Smart City scenarios or updating entire fleets of vehicles. This KPI evaluates the efficiency and scalability of multicast communication and mass updates.

ID	FR.FR.3 (Mandatory)
Title	**Device status auditing**
Actors/Components Involved	Blockchain Infrastructure, Secure Oracles, Off-chain Data Storage, Attestation agent,
Description	**Background:** The need for data auditing is not new, while there is an exhaustive literature with works for blockchain-based data auditing for different domains and applications. Regular audits can foster the identification of potential security risks and device status auditing can assist to maintain a high level of security. Thus, the continuous and automated auditing, including the device statuses, in a secure and trusted way to support the demanding business needs is a necessary REWIRE feature. **Description:** Blockchain technology is necessary to maintain the collected evidence, while smart oracles can be leveraged to support data collection (e.g., device statuses). Prior to any action, there is a need for established authentication between the device and the oracle. The smart oracles can filter, format, and perform a veracity check on the submitted data before storing them on-chain. On top of that, oracles are necessary to be TEE-enabled in order to overcome the data veracity issue and assure that the audit mechanism is trustworthy itself. Thus, all the stored information can be used as traceable and non-tampered evidence. Device status auditing is an essential aspect of the REWIRE project as it enables continuous monitoring of the security status of IoT devices. This function helps in identifying potential vulnerabilities in the system, thus reducing the risk of cyber-attacks. More specifically, the "health" status of a device or critical system should be auditable (also for certification purposes). Thus, results from the attestation of devices or from the SW/FW validation process should be available to other actors through the blockchain infrastructure. Collected data should reflect the

	trustworthiness of the systems. For example, when setting up communication with another device or a device wants to enter the overall infrastructure network (secure device onboarding), the status of that device before proceeding should be in a trusted state. By regularly auditing the device status, the project can ensure that the hardware and software components are functioning as intended, and no malicious activity is occurring. This can also result in a form of certification, demonstrating compliance with industry standards and regulations. **Remarks:** – The decentralised smart oracles enable data collection through smart contracts in a secure and trustworthy manner so that the security audit itself is trustworthy. – The oracles will be supported and embedded with a TEE so that it can run secure operations within a trusted and safe environment.
Technology enablers	Hyperledger Fabric, Keystone
Connected To Other Requirements	• **FR.FR.7**: **Zero Touch device Onboarding**: A device status auditing mechanism mandates secure device onboarding. • **FR.FR.12**: **Trust Aware Continuous Authentication and Authorisation**: A secure and trusted device status auditing mechanism mandates a continuous authentication and authorisation of the device. • **FR.FR.14**: **Data veracity management**: A core research challenge that needs to be addressed is the veracity of the collected data since it is necessary for valid auditing. • **FR.FR.16**: **AI-assisted misbehaviour detection**: The AI-misbehaviour detection mechanism can be a potential source of data that reflect the security level of a device. Such kinds of data are transmitted through the oracles. • **FR.FR.19**: **Device provenance and device status**: Device provenance and status is necessary for the corresponding auditing.
Impactful Attacks and Mitigation Measures	Device tampering, malware attacks and insider attacks should be able to be identified during the device status auditing of REWIRE. Implementing a combination of hardware and software-based security mechanisms, using secure communication protocols, and conducting regular device inspections and scans are crucial to mitigate the aforementioned attacks.
KPIs	• Block time creation • Average transaction time

ID	FR.FR.4 (Mandatory)
Title	**Application and security data event sharing**
Actors/Components Involved	Blockchain Infrastructure, Secure Oracles, Off-chain Data Storage
Description	**Background:** Sharing application and security events enables quick and crucial decision making about the security level and potential countermeasures against cyberattacks. However, the current event sharing solutions do not allow easy communication and knowledge sharing among detection systems exploiting AI-assisted detection techniques. This type of data is highly sensitive and must be protected. Thus, both confidentiality and integrity are mandatory requirements. However, due to the dynamic nature and the amount of event data a centralized way is not an option, while decentralised solutions (e.g., Blockchain) are not always scalable. **Description:** REWIRE framework should be able to identify misbehaviour of

	devices and report threat intelligence data generated, in the form of events. Data generated in the context of threat Intelligence actions (e.g., attestation, SW/FW validation, AI-based misbehaviour detection) shall be made available to the administrator of the monitored environment, (acting as the security service operator), to be able to dynamically assess its security posture and take mitigation actions. In the context of REWIRE, these application and security event data are stored on a Blockchain platform for security, transparency, immutability, and decentralization qualities. However, application and security data may be of high volume and/or velocity and stem from different layers of a system architecture (application, network, etc.). That is, a Blockchain infrastructure needs to operate in tandem with external (off-chain) data storage solutions to maintain bulky data and have a synchronised operation for data indexing with the Blockchain per se. On top of that, this functional requirement could enhance the runtime monitoring capabilities of REWIRE by a) assisting the administrator to take informed decisions on misbehaving devices and act, b) sharing of threat Intelligence data within REWIRE distributed environment will allow other entities (e.g., devices) operating in the same use case to understand the security status of other devices they need to interact with. **Remarks:** – The AI-based misbehaviour detection needs to provide an output that will enable the system administrator to promptly identify threats/attacks and utilise the results for risk assessment and further mitigation/corrective actions. – Blockchain provides secure, transparent, immutable, and decentralised storage for threat intelligence data; nevertheless, due to high data volume and velocity, a combination of blockchain and off-chain storage solutions is required for efficient data management and synchronisation.
Technology enablers	Hyperledger Fabric, AI-assisted detection, Secure Oracles
Connected To Other Requirements	• **FR.FR1: Dynamic awareness on potential vulnerabilities and threats and complete overview of the deployed environment**: Application and security data events needs be considered on the risk assessment methodology of REWIRE.
Impactful Attacks and Mitigation Measures	Sibyl, DDoS and insider attacks should be able to be identified as security events and shared through the blockchain infrastructure. Thus, proper mitigation action can be applied upon such an event is detected.
KPIs	• Average Processing time • Total latency time

ID	FR.FR.5 (Mandatory)
Title	**SW/FW unpacking and vulnerability analysis**
Actors/Components Involved	SW/FW Validation Component
Description	**Background:** SW//FW verification assesses whether a product is built correctly. Thus, a FW/FW shall be validated before the deployment on a device, either during the design phase or during the runtime phase of the framework and before the over the air SW/FW update. Detecting SW/FW vulnerabilities before the deployment avoids unnecessary risks during runtime and is of paramount importance in REWIRE. **Description:** In the context of REWIRE, the validation is part of an end-to-end chain covering device integrity and trust. Guided fuzzing and symbolic execution techniques are combined to verify whether SW/FW contains any vulnerabilities. In REWIRE, using a combination of static symbolic execution and fuzzing this analysis

	aims to find crashing inputs for the targeted FW/SW. The SW/FW can be emulated on a system dedicated to the software analysis, in order to easily scale it to many different devices and firmware versions. Through the use of root cause analysis, the information on these crashing inputs can be used to determine the existence of bugs in the firmware. When feasible, these bugs will be analysed further to determine whether they constitute a vulnerability. The information on identified vulnerabilities can then be communicated to other components in the REWIRE project, such as the risk assessment tool for increasing the awareness of the security operator, or, in the context of the Design-time phase of the REWIRE framework, this information can be given as input to the formal verification tools in order to analyse the overall security of the HW/SW co-design. **Remarks:** – The SW/FW package must be accessible by a third party, i.e., it cannot be encrypted or packaged using a proprietary format that is not publicly known. – The firmware has to be based on a Unix-like operating system and compiled for a RISC-V architecture. – The analysis combines dynamic and static analysis techniques to combine their strengths and compensate for their weaknesses
Technology enablers	Fuzzing, Symbolic Execution, Emulation, Root cause analysis
Connected To Other Requirements	• **FR.FR.2**: **Secure remote asset management and reconfiguration effectiveness**: FR.FR.2 refers to the need for over the air updates. Thus, before any deployment of a new FW/SW, the SW/FW unpacking, and vulnerability analysis needs to take place to guarantee that the SW/FW is vulnerability-free. • **FR.FR.1 Dynamic awareness on potential vulnerabilities and threats and complete overview of the deployed environment:** The Risk Assessment needs to take as an input the vulnerability analysis performed during SW/FW unpacking.
Impactful Attacks and Mitigation Measures	Attacks resulting from software-related vulnerabilities such as buffer overflows, use-after-free vulnerabilities, integer overflows, etc. should be able to be identified during SW/FW unpacking at runtime. These attacks can be mitigated by denying access to the affected service(s) until such time as the vendor has deployed a patch fixing the vulnerability.
KPIs	• Detection accuracy • Speed of detection

ID	FR.FR.6 (Mandatory)
Title	**Efficient device state monitoring**
Actors/Components Involved	REWIRE Tracer, SW/FW Validation Component (adding specific hooks for the introspection of applications)
Description	**Background:** The increasing threat surface of IoT devices mandates to consider the underline security risks and to ensure that those devices are secure and trusted. REWIRE mechanisms should be able to identify critical states of a systems' operation, (e.g., specific signals in ECUs) in order to check for the correct operational behaviour of devices. **Description:** In the context of REWIRE, the addition of monitoring hooks will be performed using the SW/FW validation component. SW/FW can be unpacked, resulting in at least a file system. The software contained in this file system can be disassembled and lifted to an Intermediate Language, on which it is more feasible to reason about the software and adjust it. In this representation, monitoring hooks will be added, after which it can be compiled back to its original architecture. The software in the unpacked firmware will be overwritten using this modified version,

	after it will be repacked. This modified version of the firmware can then be installed on a device, resulting in live monitoring of its behaviour. Note that installing modified SW/FW is not possible on a device where creating legitimate firmware is intentionally made more challenging, such as with proprietary packaging formats, encrypted firmware, or signed firmware. After the deployment of a SW/FW, the injected monitoring hooks will enable the REWIRE tracer to identify efficiently critical states of a systems' operation, in order to check for the correct operational behaviour of devices in the context of the REWIRE attestation schemes. **Remarks:** – The firmware package must be accessible by a third party, i.e., it cannot be encrypted or packaged using a proprietary format that is not publicly known. – Monitoring hooks are applied statically, meaning the firmware image itself is changed.
Technology enablers	Disassembling, Lifting, Compilation, (Un)packing of firmware
Connected To Other Requirements	• **FR.FR.3**: **Device status auditing**: REWIRE mechanisms should be able to identify critical devices' statuses for checking for the correct operational behaviour of devices. • **FR.FR.18**: **Secure Measurement and Attribute Extraction**: REWIRE measurements are necessary checking the device state.
Impactful Attacks and Mitigation Measures	SW/FW attacks should be able to be identified during monitoring. The increasing threat surface of IoT devices mandates the IoT devices' monitoring to mitigate this surface.
KPIs	• Percentage of firmware that can be successfully monitored.

ID	FR.FR.7 (Mandatory)
Title	**Zero Touch device Onboarding**
Actors/Components Involved	Keystone TEE, Blockchain Infrastructure, Secure Oracles, MUD Profile Server, Privacy CA, Domain Manager, Attestation Agent
Description	**Background**: Faced with the rapid increase in IoT inter-connected devices and the high demand for new services, the management of such devices is getting complex. These devices are subject to an expanding list of attacks that exploit both software vulnerabilities and design choices, highlighting the importance of management cryptographic keys. In addition, traditional onboarding approaches cannot keep up with rapidly evolving application requirements. The limitations of traditional onboarding models, such as security and time-consuming manual steps, especially in the IoT domain led towards developing zero-touch onboarding mechanisms (also known as zero touch provisioning). **Description**: REWIRE needs an efficient and scalable zero-touch onboarding solution to configure the IoT devices onto a network as well as a minimized attack surface. Also, an access control mechanism is needed to assist in the onboarding and managing of devices into the network ensuring secure (authenticated) enrolment and consistent standardisation. More specifically, in REWIRE an Attribute-based Access Control (ABAC) along with the trust-related attributes in the form of VPs is considered a natural fit to adopt. The cryptographic key used to generate the VP is held in the TEE (utilising the REWIRE key management mechanisms), with key usage policies restricting its use, resulting in control over whether the device can get access to the network, given that it exhibits specific attributes advocating its operational correctness. Also, a proposed solution that will be investigated further in REWIRE is the authentication protocol using DAA (instead of "group" keys to preserve the privacy of authenticators). Recall, DAA is a cryptographic scheme used to ensure security and privacy guarantees and it has been adopted by TGC TPM2.0 for Elliptic Curve

	based DAA (ECDAA). REWIRE will focus on DAA with attributes (DAA-A) protocol, in which the platform can select which attributes (from VPs) are shown/hidden to the verifier. The authenticity of the hidden attributes will be proved by a zero-knowledge protocol and usefully DAA-A provides user-controlled linkability. **Remarks**: – Currently, the DAA protocol is not included in the W3C description of VCs. – The two most prominent EC based DAA protocols for TPM 2.0 can be extended to DAA-A protocols.
Technology enablers	Keystone TEE, Hyperledger Fabric, Verifiable Presentations, DAA, ABAC
Connected To Other Requirements	• **FR.FR.25**: **Provably secure crypto protocols and algorithms**: All exchanges of sensitive data such as access control and DAA solutions should be protected through cryptographic means. • **FR.FR.22**: **Policy-based device state configuration**: During onboarding, policies regarding the device state are necessary to fulfil the security requirement. • **FR.FR.9**: **Dynamic Credential Management**: Dynamic credential management is necessary for the authorisation during onboarding. • **FR.FR.11**: **Flexible and reliable key management**: Key management is necessary for providing the necessary crypto primitives to perform the secure onboarding solution and the management of the cryptographic keys. • **FR.FR.17**: **Common Trusted Computing Base**: A common TCB is necessary to be adopted in order to fulfil the correct device state and hence the secure onboarding. • **FR.FR.19**: **Device provenance and device status**: Zero-touch onboarding is tightly coupled with the device provenance and status.
Impactful Attacks and Mitigation Measures	REWIRE's zero touch provisioning is tolerant to Man-in-the-middle attacks as well as any device's misconfiguration. In addition, REWIRE minimizes the attack surface by adopting key restriction usage policies, for allowing only devices at correct state to access their cryptographic keys, and DAA techniques, for privacy preservation of the device.
KPIs	• Elapsed time to complete a secure device onboarding. • Scalability of the device onboarding engine w.r.t the number of devices engaged.

ID	FR.FR.8 (Mandatory)
Title	**Continuous tracking and auditability of patch management**
Actors/Components Involved	Attestation Agent, Blockchain Infrastructure, SW/FW Validation Component
Description	**Background:** Due to scalability in massive deployments of IoT devices, the rollout of new SW/FW in a massive scale implies that several updates will take place in parallel. Network administrators require a human-friendly summary whenever a device finished successfully and securely the SW/FW update. Thus, continuous monitoring and patch management to increase cyber resilience in a secure way is of paramount importance in REWIRE. **Description**: In the context of REWIRE, devices need to be updated through a decentralized FUOTA process supported by a Blockchain Infrastructure, enabling on-demand security updates and patches. The process and the stages of the update will be logged in the Blockchain for auditing purposes. i.e., the validation result of the SW/FW, the signature of a SW/FW and the deployment, the SW/FW details. In a nutshell, the service that deploys the SW/FW update needs to verify that the

	correct SW/FW is installed. Thus, the administrator should be able to track the SW/FW version installed in each device of its deployment. This is to ensure that REWIRE SW/FW is up-to-date, secure, and compliant with organizational policies, minimizing the risk of security breaches and maximizing at the same time system performance. On top of that the MUD standard from the Internet Engineering Task Force (IETF) will be also integrated as part of the process to configure/reconfigure the device in a secure way. Throughout its lifecycle, the device could be reconfigured to accommodate it to the changing threat landscape or requirements. Indeed, even the original manufacturer of the device may publish secure updates and accordingly modify the MUD file when necessary. **Remarks**: – SW/FW update will be triggered by the administrator based on the risk assessment output.
Technology enablers	MUD, Hyperledger Fabric
Connected To Other Requirements	• **FR.FR.7**: **Zero Touch device Onboarding**: Continuous tracking and auditability of patch management is crucial to achieve secure device onboarding.
Impactful Attacks and Mitigation Measures	REWIRE will not only provide a scalable mechanism for SW/FW update but will also ensure the correct execution of the patching mechanism itself in a secure and validated way.
KPIs	• Scalability of the patch management w.r.t the number of devices engaged.

5.2 Security Requirements

ID	FR.FR.9 (Mandatory)
Title	**Dynamic Credential Management**
Actors/Components Involved	TEE, Attestation Agent, Blockchain Infrastructure.
Description	**Background**: Most existing credential management solutions require a centralized authority, either for attribute registration or credential verification. On the contrary, SSI enables entities to have ownership and control of their unique identity data without involving any third party. Currently, several works are exploring how to leverage blockchain to build SSI solutions, however, there is a lack of systematic architecture design for blockchain-based SSI systems. On top of that, current solutions are considered coarse-grained and may underperform or lack on data security (e.g., inadequate access control for credentials). In addition, highly distributed environments mandate the continuous authorisation and authentication of devices as well as secure device onboarding. Apart from the certificates used to verify the identity, certificates are also employed to verify the data provenance; a fact that adds further complexity to the system. This highlights the need in REWIRE of credential management (including revoking credentials in case of misbehaviors) in a dynamic way. **Description**: REWIRE will generate credentials for ensuring efficient, trustworthy, and secure communication between entities. Trust claims are made using VPs, while a selective disclosure of attributes will be used in order to minimise the information which needs to be exposed by the attributes to the verifying entities. More specifically, VPs are comprised of self-issued verifiable credentials and the key used to generate the VP is stored in the TC wallet. In the context of REWIRE, dynamic credentials are necessary as they can be generated on-demand, they are unique to a client (instead of static credentials which are pre-defined and shared) and they can change over time - a credential is associated with some policy encompassing the expiration date of when the credential needs destroying.

	Crucial to security, the dynamic functionality also mitigates security risks if a credential is leaked. In this case, the credential is revoked and remediated. To do so, REWIRE necessitates dynamic attestation which is reflected in the dynamic nature of credentials. Last but not least, blockchain technology will be also used to facilitate sharing and auditing of verifiable credentials between entities that want to establish a trust relationship in a zero-trust manner. **Remarks**: - VPs issued by parties have the following advantages: interoperability (DIDs are "chain-agnostic" so they're not permanently bound to a single BC); privacy (communicating parties can issue verifiable statements with the selective disclosure of attributes (DAA-A)); scalability (maintaining a subset of credentials off-chain can reduce the costs). - Credentials used for static attestation would only demonstrate the state of a device at the time of executable code, therefore, it would not provide a view of program behaviour at run-time.
Technology enablers	Keystone, Hyperledger Fabric
Connected To Other Requirements	• **FR.FR.15**: **Confidentiality and integrity in data processing**: Dynamic credential management is of paramount importance to ensure the data integrity and confidentiality during processing (e.g., by utilising ABE). • **FR.FR.12**: **Trust Aware Continuous Authentication and Authorisation**: The dynamic credential management is the baseline for device authentication and authorisation. • **FR.FR.22**: **Policy-based device state configuration**: Policies are based on the credentials of the device (e.g., attributes). • **FR.FR.7**: **Zero Touch device Onboarding**: Dynamic credential management is necessary to achieve zero touch device onboarding. • **FR.FR.11**: **Flexible and reliable key management**: Key management is the baseline for providing the necessary crypto primitives (e.g., digital signatures) for dynamic credential management. • **FR.FR.19**: **Device provenance and device status**: Device status corresponds to the overall device trustworthiness and is reflected in the device credentials (e.g., attributes). • **FR.FR.17**: **Common Trusted Computing Base**: A common TCB is the baseline as the trust anchor for credential management. • **FR.FR.25**: **Provably secure crypto protocols and algorithms**: Dynamic credential management is necessary to ensure the security of crypto protocols and algorithms.
Impactful Attacks and Mitigation Measures	Dynamic credential management prevents unauthorised devices to join the REWIRE network. Also, ABE protects data confidentiality and tampering. A revoking mechanism is crucial for excluding a device that misbehaves in the network.
KPIs	• Scalability of the credential management w.r.t the number of devices engaged and the number of attributes. • Signing and verification times for the VPs

ID	FR.FR.10 (Mandatory)
Title	**Establishment of Secure and Authenticated Communication Channels**
Actors/Components Involved	SW/FW Validation Component
Description	**Background:** The establishment of secure and authenticated channel is imperative

	in REWIRE with the huge amount of heterogeneous interconnected devices. It is vital to protect the critical process of secure SW/FW update and protect this process by deploying secure and authenticated communication channels. **Description:** REWIRE will consider the critical character of SW updates, as they represent the first trust level of the system, and the fact that REWIRE devices often operate in unmonitored environment, the channel should be secure even in the presence of physical attackers capable of performing side-channel attacks. Moreover, being able to achieve this security level based on off-the-shelf components and without the need for the implementer to have a deep expertise in the field of side-channel countermeasures would allow swift and easy development of REWIRE devices. Towards this direction, REWIRE needs to design an Authenticated Encryption (AE) algorithm. This algorithm ensures both the confidentiality and the authenticity of exchanges between two actors sharing a symmetric key, meaning that attackers cannot obtain any information on the transmitted payload, nor modify this payload in any way without being detected. This algorithm should also meet a state-of-the-art definition of security in adversarial physical conditions, such as CIML, precisely defining which security properties hold even if the attacker can monitor side-channel behaviour, tamper with nonces. The algorithm will be implemented as a leakage-resilient mode of operation based on a standard block cipher such as the AES. In as much as possible, leakage resilience will be enforced by the mode of operation itself, without relying on side-channel countermeasures in the implementation of the underlying block cipher. **Remarks:** – The shared symmetric key used for SW updates must be a long-term key that can of course not be exchanged nor derived using methods that are not protected against side-channels, or these methods must be applied before deployment, in a secure environment. – The resulting cryptographic algorithm will consist in: (a) A SW implementation of the mode of operation, relying on a block cipher implemented in HW. (b) A characterization of the minimal expected properties of the HW block cipher, together with an example of off-the-shelf device meeting these requirements.
Technology enablers	AES
Connected To Other Requirements	• **FR.FR.2: Secure remote asset management and reconfiguration effectiveness**: The leakage-resilient mode of operation of the AES will support the SW/FW update processes of REWIRE • **FR.FR.5: SW/FW unpacking and vulnerability analysis**: The SW/FW unpacking, and vulnerability analysis should be performed in a secure and authenticated channel.
Impactful Attacks and Mitigation Measures	Potential impactful attacks are Man-in-the-middle (MITM), cryptanalysis and side-channel attacks.
KPIs	• Performance of establishment a secure and authenticated channel.

ID	FR.FR.11 (Mandatory/Optional)
Title	**Flexible and reliable key management**
Actors/Components Involved	Keystone TEE, Security-by-design Monitors, Attestation Agent
Description	**Background**: The key management process includes the secure generation, distribution, operation, storage and deletion of cryptographic keys, while the huge number of heterogeneous IoT devices, implies complex key management for different involved stakeholders, types of devices, and key lifecycles. Encrypting data

	to protect the confidentiality of the underlying information is a necessity. When it comes specifically to the SW update key, once the corresponding encryption (symmetric) key has been generated and used for its purpose, it must be stored for later use when the corresponding ciphertext needs to be decrypted. Storage of cryptographic keys is challenging, and solutions based on key hierarchies are necessary. **Description**: Intuitively, this design in the REWIRE architecture will organise cryptographic keys so that a root (master) key encrypts so-called leaf (children) keys, with the leaf keys encrypting the data in question. One difficulty is to ensure that the master key is simultaneously secure and accessible to decrypt leaf keys. Benefits to REWIRE of utilising a key hierarchy include minimising the amount of plaintext key material requiring protection and storage – plus, storing a master key alone restricts access to just one key. Moreover, key hierarchies enable segmentation of encrypted data. That is, data can be encrypted using independent cryptographic keys which minimises the impact of potential key corruption/loss. Furthermore, the REWIRE KMS manages keys in the Security Monitor since critical services, like key derivation, must run in isolation from other SW running on the same platform. The TEE must manage keys for remote attestation, sealing, migration etc. **Remarks**: – Key hierarchies can be comprised of several layers of keys, such that a parent key provides protection to their corresponding leaf keys, all the up to the master root key which encrypts all keys irrespective of what layer they belong to.
Technology enablers	Keystone
Connected To Other Requirements	• **FR.FR.8**: **Continuous tracking and auditability of patch management**: Key management is necessary for key establishment and hence to achieve a secure SW/FW update. • **FR.FR.7**: **Zero Touch device Onboarding**: Key management is necessary to achieve zero touch device onboarding. • **FR.FR.9**: **Dynamic Credential Management**: Key management is the baseline for providing the necessary crypto primitives (e.g., digital signatures) for dynamic credential management. • **FR.FR.22**: **Policy-based device state configuration**: Key-restriction policies binded to the device state are used allow or not the key usage. • **FR.FR.17**: **Common Trusted Computing Base**: A common TCB is the baseline as the trust anchor for key management. • **FR.FR.25**: **Provably secure crypto protocols and algorithms**: Proper key management is a prerequisite to ensure the security of crypto protocols and algorithms.
Impactful Attacks and Mitigation Measures	Attacks relevant to higher risk of cryptographic key exposure due to long-term storage of data-at-rest or tampering can be mitigated with the usage of key restriction usage policies and with frequent key updates.
KPIs	• Scalability of the key management w.r.t the number of devices engaged. • Performance of the key management w.r.t the time to establish a key • Efficiency of key hierarchy construction of different types of keys

ID	FR.FR.12 (Mandatory)
Title	**Trust Aware Continuous Authentication and Authorisation**
Actors/Components	Blockchain Infrastructure, Attestation Agent, TEE

Involved	
Description	**Background:** Several well-established access control mechanisms exist in the literature. The adoption of an access control mechanism such as ABAC or RBAC for enabling authenticated and authorised resource and data accessing is necessary in the IoT domain. It must be noted also here the need for external, but authenticated and authorized, entities to retrieve data, mainly for certification purposes. The access control needs to consider the use of trust-related attributes (claims) as VPs. This requirement is needed in the REWIRE project to provide advanced security measures for the data on Blockchain Infrastructure, with the help of advanced technology enablers, such as wallet, smart contract, and TEE technologies. **Description:** In REWIRE it is necessary to ensure that the security measures implemented are not only robust but also efficient, as inefficient security measures may lead to practical difficulties and a lack of adoption. Thus, REWIRE needs to consider the use of crypto that can offer both security and efficiency. By incorporating such technology, the REWIRE project can ensure that sensitive data is protected without sacrificing usability and convenience. Besides, it's necessary to have access to reliable and trustworthy data for security assessment. This data is mainly related to the devices' statuses but also those hard copies of the data stored in an off-chain and cloud server backend. Recall that it may not be practical to store all data on-chain and it is necessary to intake updated statuses to the ledger. To ensure authenticity and integrity, a robust mechanism for authenticated off-chain data retrieval is essential. By doing so, the REWIRE project can access dependable data for security assessment. In addition, the above provided solutions will be designed in an efficient and cost-effective access control and TEE protection. This functional requirement enhances the capabilities of REWIRE by providing an advanced and efficient security mechanism that continuously monitors user behaviour and detects potential security risks in real-time. By adopting Trust Aware Continuous Authentication and Authorization, the REWIRE project can ensure that only authorized and authenticated resources can access the sensitive data. This technology also helps prevent unauthorized access and cyber-attacks, contributing to the overall security of the REWIRE project. Besides, the external data retrieval will be considered for the authentication so that the income and outcome data sources are trusted and verified. It has to be noted that the authentication and authorization mechanisms of REWIRE will not only be focusing on the devices belonging in the monitored ecosystem, but the same principles will be applied and need to be followed by external entities that need to access to the collected data (mainly for certification purposes). **Remarks:** – Flexibility related to who can generate these claims either by a trusted issuer or self-made attestation evidence is also important.
Technology enablers	Hyperledger Fabric, ABAC, RBAC, Keystone
Connected To Other Requirements	• **FR.FR.3**: **Device status auditing**: Authentication and authorisation of the device is necessary for the device status auditing. • **FR.FR.14**: **Data veracity management**: Authentication and authorisation of the device is necessary for veracity of the collected data. • **FR.FR.17**: **Common Trusted Computing Base**: Authentication and authorisation of the device is based on the underline TCB.
Impactful Attacks and Mitigation Measures	Potential impactful attacks are man-in-the-middle (MITM), insider and DDoS attacks. Implementing a combination of hardware and software-based security mechanisms as well as regularly updating passwords and implementing DDoS protection mechanisms could minimise these attacks.
KPIs	• Authentication Success Rate • Time to authorize a device • Number of crypto operations (i.e., signature encryption per second)

	• Size of signatures and certificates

ID	FR.FR.13 (Mandatory)
Title	**Data provenance**
Actors/Components Involved	TEE
Description	**Background:** The term data provenance also called data lineage, refers to a record trail that accounts for the origin of a piece of data together with an explanation of how and why it got to the present place. REWIRE devices must be able to establish and verify the origin of sensitive data they receive, and potentially record this proof of origin together with data at rest, as they run in a fully decentralized environment. **Description:** Sensitive data exchanged in the framework of REWIRE must be protected by cryptographic means, to allow verifying its provenance and the fact that it has not been modified. Depending on the context, these means can consist in: • Asymmetric cryptographic primitives, such as digital signature. Digital signatures provide the advantage of being transferrable to a third party, meaning that it is possible for an entity to prove, in an undeniable way, that a piece of data was signed by another entity. Signatures can also be chained if multiple provenances (e.g., along a transmission path) must be established. However, digital signatures are more expensive than symmetric primitives, and signature generation is more difficult to protect against side-channel attacks (signature verification, on the other hand, does not suffer from this drawback, as it involves no secret parameter). • Symmetric cryptographic primitives, such as authenticated encryption. Symmetric primitives allow verifying that a piece of data originates from an entity knowing a specific secret key and has not been modified. However, this verification is not transferrable to a third party (no distinction can be made between the data sender and receiver) and can typically not be chained. Less costly and easier to protect against side-channel attacks, they remain nonetheless useful in cases where such transfer is not required. **Remarks:** – Data provenance (ability to check the integrity of data and the identity of its emitter) must not be confused with verifiability that the data is harmless, which is addressed by misbehaviour detection.
Technology enablers	Keystone, Hyperledger Fabric
Connected To Other Requirements	• **FR.FR.11**: **Flexible and reliable key management**: Key management is the baseline for providing the necessary crypto primitives (e.g., digital signatures) for assuring the integrity and data provenance. • **FR.FR.17**: **Common Trusted Computing Base**: Data Provenance is based on the underline TCB.
Impactful Attacks and Mitigation Measures	Potential impactful attacks are man-in-the-middle (MITM), cryptanalysis and side-channel attacks.
KPIs	• Scalability of provenance w.r.t the number of devices engaged, several actors and data.

ID	FR.FR.14 (Mandatory)
Title	**Data veracity management**
Actors/Components	Blockchain Infrastructure, Secure Oracle, Attestation Agent, Keystone TEE

Involved	
Description	**Background:** Data veracity management is a critical aspect of ensuring the integrity of data within the IoT domain. By establishing a trustworthy and auditable data provenance mechanism, the system can ensure the veracity and traceability of vital data, thereby augmenting the security and operational assurance of the IoT ecosystem. A blockchain-based oracle can provide data veracity validation by verifying the accuracy and integrity of data before it is entered into the blockchain. On top of that, the execution of data transaction needs to take place through these secure oracles and with the use of smart contracts supported by root of trust. **Description:** To provide trusted and accurate data, the use of a blockchain-based oracle is essential. In the context of REWIRE, a secure oracle acts as a trusted intermediary between external data sources and the blockchain infrastructure. The secure oracle is responsible for verifying the authenticity and veracity of data before it is entered into the blockchain. The use of a blockchain oracle enhances the veracity of the data management process and helps to prevent the introduction of fraudulent or inaccurate data into the system. This functional requirement enhances the capabilities of REWIRE by providing a reliable method for ensuring the veracity of data within the system. By utilizing a TEE based blockchain oracle, the REWIRE project can ensure that only trusted and accurate data is utilized for security assessment and certification. This helps to improve the overall security posture of IoT devices and ensures that they meet the necessary cybersecurity standards. **Remarks:** – Secure oracles are enabled by the REWIRE customizable TEE to offer the necessary crypto primitives and ensure the secure management of data sharing.
Technology enablers	Keystone, Hyperledger Fabric, Secure Oracle
Connected To Other Requirements	• **FR.FR.12**: **Trust Aware Continuous Authentication and Authorisation**: C continuous authentication and authorisation of the device is a prerequisite for data veracity. • **FR.FR.3**: **Device status auditing**: REWIRE mechanisms should be able to identify critical devices statuses for checking the correct operational behaviour of devices which also affects the data veracity. • **FR.FR.17**: **Common Trusted Computing Base**: Data veracity is based on the underline TCB.
Impactful Attacks and Mitigation Measures	Potential impactful attacks are device tampering, sybil attacks and insider attacks. Implementing a combination of cryptographic mechanisms, access controls, monitoring solutions, and regular audits could minimise these attack vectors.
KPIs	• Data Integrity Rate • Data Consistency Rate

ID	FR.FR.15 (Mandatory)
Title	**Confidentiality and integrity in data processing**
Actors/Components Involved	TEE
Description	**Background:** REWIRE needs to ensure the data integrity and confidentiality of critical functions running on the IoT devices so that to guarantee the operational assurance of the device and the critical services offered by both the device and the REWIRE platform. Such critical functions can include key management, attestation or SW update validation. This must be performed in a way that an application built with trusted and untrusted parts is able to call one of such critical functions from the untrusted part, then through a high-privileged component transition data and execution to the trusted part and remain unable to access the trusted part. This

	guarantees confidentiality and integrity. On the other hand, the trusted part must be able to execute code and manipulate data and return results back to the untrusted world through the high-privileged component. It is also important to be able to seal and unseal data, while having a way, such as monotonic counters, which prevents rollback attacks and bad data updates. **Description:** The key component to achieve this is the REWIRE TEE (for which, namely, Keystone has been proposed). By means of the capabilities normally issued in Keystone, REWIRE must ensure the integrity and confidentiality of data passing through the untrusted world to the trusted world of the device. This requirement needs to be considered also on the level of confidentiality of data between different TEEs, between trusted portions of memory (i.e., enclaves), between the OS and the enclaves, between user applications and the enclaves, plus in terms of the necessity of being able to successfully migrate the execution (together with data) of critical functions between different TEEs. The REWIRE TEE must protect the confidentiality and integrity of all the enclave code and data at all points during execution. Ideally, it must protect against a physical adversary even though it has access to DRAM or the OS. The high-privileged component must be fully trustable, and it must be verified by the root of trust in hardware during boot time. Such high-privileged component (i.e., the Security Monitor if referring to Keystone) must perform memory isolation (e.g., by means of leveraging PMP in the case of the RISC-V ISA), while being able to receive calls from the OS and such isolated memory regions. **Remarks:** – A trusted high-privileged component must specify physical memory regions to isolate and control their memory access permissions. – Complete isolation must be achieved from all other elements except the high-privileged component and the own isolated memory region.
Technology enablers	Keystone
Connected To Other Requirements	• **FR.FR.2**: **Secure remote asset management and reconfiguration effectiveness**: The SW/FW update provenance is computed inside an isolated memory region with the support of the TEE. • **FR.FR.3**: **Device status auditing:** Confidentiality and integrity is crucial for auditing which also includes attestation results. • **FR.FR.11**: **Flexible and reliable key management**: Key management is the baseline for providing the necessary crypto primitives for assuring confidentiality and integrity, while the TEE can allow the key generation take place in enclaves. • **FR.FR.13**: **Data provenance:** Confidentiality and integrity is crucial for data provenance. Also, the TEE can allow intra-enclave enhancement of data provenance.
Impactful Attacks and Mitigation Measures	Keystone does not natively protect the enclave or the SM against timing side-channel attacks. SM, RT, and app programmers should use existing software solutions to mask timing channels. In addition, Keystone hardware manufacturers can supply timing side-channel resistant hardware
KPIs	N/A

5.3 Operational Assurance Requirements

ID	FR.FR.16 (Mandatory)
Title	**AI-assisted misbehaviour detection**
Actors/Components Involved	Blockchain Infrastructure, Secure Oracles, AI-based Threat Intelligence, Off-chain Data Storage
Description	**Background:** The correct operation and behaviour of every device in a network needs to be ensured to provide the users with secure services and protect the

	integrity of the data that is shared across the nodes of the distributed network. Misbehaviour detection can offer an automatic method to detect and address threats by checking the plausibility, consistency, and behaviour of these nodes and on the network as a whole. AI-assisted misbehaviour detection mechanisms are found to be more effective against attackers that pretend to be a network node compared to traditional security mechanisms. **Description:** REWIRE will leverage AI-assisted misbehaviour detection mechanisms that will help identify abnormalities by checking the nodes' behaviour to align with protocol specifications. They will also evaluate the correctness of the data that is shared across the devices. The AI-misbehaviour detection mechanism will utilize both business and network data originated from the Blockchain and off-chain data storage to detect abnormalities. It will then provide the appropriate information about the detected anomalies in order to assess the level of threat and take the actions to protect the system against them. **Remarks:** – Blockchain will be used for AI-based misbehaviour detection as part of the chain code of SCs to ensure the trustworthy training and inferencing of AI FL-models.
Technology enablers	Hyperledger Fabric, FL-models
Connected To Other Requirements	• **FR.FR.1**: **Dynamic awareness on potential vulnerabilities and threats and complete overview of the deployed environment**: The AI-misbehaviour detection mechanism needs to provide information regarding misbehaviour incidents for a more accurate and dynamic risk assessment. • **FR.FR.3**: **Device status auditing**: The AI-misbehaviour detection mechanism needs to provide information regarding misbehaviour incidents for auditing of the device.
Impactful Attacks and Mitigation Measures	The AI-misbehaviour detection will be used to detect abuse case against the monitored systems of the ecosystem where REWIRE framework is deployed. These abuse case may be a product a malevolent actor, or they may be attributed to system misconfigurations. Any attack which is reflected to the data which are used in the AI system could be a potential case where the AI-based misbehaviour detection mechanisms of REWIRE can be used. Our aim will be to identify attacks at the application layer of the monitored deployments focusing on data logs and behavioural characteristics of the devices. The AI-based misbehaviour detection will focus on the detection of incidents. Mitigation measures for attacks could be applied as attestation policies that can regulate the behaviour of specific processes that exhibit a malicious behaviour.
KPIs	• Detection accuracy • Timeliness of detection

ID	FR.FR.17 (Mandatory)
Title	**Common Trusted Computing Base**
Actors/Components Involved	TEE
Description	**Background:** The security of modern computing systems has come under scrutiny due to the abundance of vulnerabilities related to the high complexity of OSs and hypervisors. Due to this, it has become more attractive to rely on smaller and lower layers, i.e., FW or even immutable HW, to enforce security and to reduce the underlying TCB. Also, due to the various and diverse systems, a common TCB integrated in all the different use cases is necessary. Briefly, the TCB of a device is the software stack and hardware components that are required for it to function correctly and guarantee the security and operational requirements. A common TCB ground can ensure the applicability of the formal verification and the security-by-design principle of REWIRE and thus its crucial for

	the project. In addition, a common base enables the integration to the various endpoints in REWIRE (e.g., oracles, end user device and back-end). **Description**: In REWIRE, the TCB solution should be used in all the diverse and safety-critical systems of the project. A common TCB will foster the evaluation the defined theorem proofs and the crypto implementations. Such a requirement leads us to the adoption of an open-source framework for RISC-V for all the systems that are going to be used in the REWIRE use cases. On top of that, by being agnostic of the underline TCB, REWIRE seeks to foster collaboration and innovation within the IoT ecosystem, allowing more flexible and adaptable solutions as the technology evolves. Thus, it is a strategic decision that ensures the future-proof of REWIRE advancements in the safety-critical IoT systems. **Remarks**: – REWIRE should aim to create a TCB that can be used in all the diverse systems of the project. – A common TCB will reinforce the assurance level as the formally verified underline technology among the different systems.
Technology enablers	Keystone, RISC-V, Genesis 2 board, Linux OS, Crypto co-processor,
Connected To Other Requirements	• **FR.FR.7**: **Zero Touch device Onboarding**: A common TCB is necessary to achieve zero touch device onboarding. • **FR.FR.9**: **Dynamic Credential Management**: A common TCB is the necessary trust anchor for credential management. • **FR.FR.11**: **Flexible and reliable key management**: A common TCB is the necessary trust anchor for key management. • **FR.FR.12**: **Trust Aware Continuous Authentication and Authorisation**: A common TCB is the necessary trust anchor for continuous authentication and authorisation. • **FR.FR.13**: **Data Provenance**: A common TCB is the necessary trust anchor for data provenance. • **FR.FR.14**: **Data veracity management**: A common TCB is crucial for the data veracity of the collected data. • **FR.FR.20**: **Operational assurance and configuration integrity**: A TCB acts as the trust anchor for assuring correct operation and configuration integrity of a device. • **FR.FR.21**: **Chain of trust creation**: A TCB acts as the trust anchor for trust establishment. • **FR.FR.26**: **Computer-aided verification of security/safety profiles of systems**: A TCB acts as the trust anchor for verification of security/safety profiles of systems.
Impactful Attacks and Mitigation Measures	The attack surface should be minimised considering the formally verified TCB.
KPIs	N/A

ID	FR.FR.18 (Mandatory)
Title	**Secure Measurement and Attribute Extraction**
Actors/Components Involved	REWIRE Tracer, SW/FE Validation Component (Hook instrumentation), Attestation mechanisms (Integrity Verification), REWIRE Tracer
Description	**Background:** Traditional static analysis techniques cannot capture the dynamic threats manipulating dynamically the execution flow of a code base. Thus, more

	sophisticated, and dynamic techniques are need, while legacy dynamic tracing techniques are not scalable due to the state explosion problem. Remote Attestation (RA) techniques are an essential part of trusted computing. Current RA techniques should provide proofs of static and dynamic system properties to offer holistic security, through verification of system's integrity. These dynamic properties are used as evidence to verify the system integrity during runtime. To do so, runtime monitoring of the system data and its execution flows enables the collection of useful information about the behaviour of the system. Thus, potential anomalies or deviations from intended behaviour can be identified. However, dynamic attestation has some challenges from the dynamic nature of the attributes itself making it difficult to identify and deduce their good state to scalability and efficiency issues. **Description**: REWIRE requires a non-intrusive runtime validation process of the runtime data and execution stream. The dynamically collected execution traces on the devices (e.g., REWIRE evidence collectors), upon a runtime attestation fail (i.e., attack indication), will be further analysed dynamically to emulate the problematic execution path and potentially uncover a previously unseen malicious execution behaviour. The REWIRE Tracer should be able to extract the system state to be validated. Apart from binary and configuration files signatures, the tracer needs to be able to perform non-intrusive memory introspection to enable control flow attestation during run-time. **Remarks**: – REWIRE, in the case where the tracer is not positioned in the trusted world (e.g., enclave), will assure that the tracer is in a secure state and the captured traces are valid and not tampered with (i.e., Integrity of Trust Measurements). – REWIRE tracing is aimed to be efficient since only a fraction of the execution needs to be monitored, due the minimisation of the stack that needs to be introspected as a result of the formal verification of the HW/SW co-design, and as a result of the use of monitoring hooks.
Technology enablers	eBPF, pTrace, RISC-V
Connected To Other Requirements	• **FR.FR.20**: **Operational assurance and configuration integrity**: The secure measurement and attribute extraction is the baseline to provide operational assurance through attestation.
Impactful Attacks and Mitigation Measures	REWIRE will assure the correct state of the tracer, either running in the enclave (trusted word) or not (untrusted word) minimising the attack surface of not valid traces.
KPIs	• More efficient and scalable tracing techniques compared to the traditional ones due to the reduced ISA space

ID	FR.FR.19 (Mandatory)
Title	**Device provenance and device status**
Actors/Components Involved	Attestation Agent, TEE, BC Infrastructure,
Description	**Background**: To ensure the overall trustworthiness of the IoT ecosystem, data are collected and analysed in order to detect potential misbehaviours. This involves management of provenance information especially in distributed environments such as IoTs. Decentralised environments must establish the origin of data (data provenance) and verify it to take a proactive approach to security rather than a reactive approach post-attack. Further, device provenance provides guarantees that a device has been onboarded correctly in the system. Moreover, the health (status) of a device is equally important in the context of REWIRE. Device status corresponds also to the results from device attestation and/or SW/FW update

	validation, considering corresponding policy regulations, and it is used to determine the device and the systems overall trustworthiness. **Description**: It is difficult to have a high level of certainty about where sensitive data might be located and the origin of data in the fully decentralised environment that is REWIRE. Thus, we need to differentiate between the integrity and attribution of a message from the verifiability of the message, which is captured for the requirement of misbehaviour detection, to determine data provenance. Tracking the origin of data may cause privacy issues, thus, where appropriate we should consider using digital signatures and DAA-A in REWIRE (see the zero-touch onboarding requirement). In doing so, we can enable the verification of authorship of verifiable statements to belong to a device without breaching the privacy of the device. With respect to device status in REWIRE, the health of a device is useful information to actors in the BC infrastructure to determine the devices' trustworthiness when attempting to onboard into the network or set up communication with other devices in the network. Device status also aids in REWIRE identifying security risks/vulnerabilities to reduce the chance of attacks. **Remarks**: – The TEE (Keystone) can allow intra-enclave enhancement of data provenance.
Technology enablers	Keystone, Hyperledger Fabric
Connected To Other Requirements	• **FR.FR.3**: **Device status auditing**: Device provenance and status is necessary for the device status auditing. • **FR.FR.7**: **Zero-touch onboarding**: Zero-touch onboarding is a perquisite for the device provenance and status. • **FR.FR.18**: **Secure Measurement and Attribute Extraction**: Secure measurement and attribute extraction is of paramount importance to verify the device status.
Impactful Attacks and Mitigation Measures	REWIRE will assure the system's integrity and detection of any device misbehaviour and/or tampering attack. Control-flow attestation.is the necessary technique for this assurance.
KPIs	• Efficient and scalable attestation techniques verifying the device status in resource constrained edge devices.

ID	FR.FR.20 (Mandatory)
Title	**Operational assurance and configuration integrity**
Actors/Components Involved	Risk Assessment, Attestation Agent, Trust Aggregation Overlay, REWIRE Evidence Collector
Description	**Background:** Several attestation schemes exist in the literature covering all phases of a device's execution; from the trusted boot and integrity measurement of an edge device, enabling the generation of static, boot-time or load-time evidence of the system's components correct configuration (e.g., Configuration Integrity Verification), to the runtime behavioural attestation of safety-critical components of a system providing strong guarantees on the correctness of the control flow (Control-Flow Attestation). However, due to scalability and performance issues of these schemes, there is a need to increase the efficiency necessary. Assurance of the operation and configuration integrity are crucial; thus the SW/FW update needs to be attested in an efficient way. **Description**: REWIRE requires a new breed of efficient attestation mechanisms that can provide the same level of security guarantees, while increasing the efficiency due to the limited vector of properties that need to be attested. Based on this attestation assessment the state of the component could be derived and classified

	into compromised, under-attack, out-dated, normal operation, and others. On top of that, REWIRE's TEE provides the necessary mechanisms to enable the isolated execution of sensitive functions in enclaves. This can be achieved by utilising Keystone's Security Monitor (SM), that ensures physical memory isolation for said enclaves by manipulating RISC-V PMP registers. Critical services, including cryptographic services like key derivation must run in isolation from any other software, including privileged one, are running on the same platform. **Remarks**: – REWIRE state and/or integrity system verification acts as an ex-ante requisite in many situations, e.g., prior to a TEE Migration but also as an ex-post requisite for others, e.g., after an OTA FW update. – The REWIRE TEE must ensure that the memory assigned to a critical service is inaccessible from other services and even the OS. – REWIRE attestation is efficient since it focuses only on sequence of assembly commands from the ISA instruction set of the RISC-V.
Technology enablers	Keystone
Connected To Other Requirements	• **FR.FR.17**: **Common Trusted Computing Base**: A common TCB is the cornerstone for assuring correct operation and configuration integrity of a device. • **FR.FR.18**: **Secure Measurement and Attribute Extraction**: The secure measurement and attribute extraction is the baseline to provide operational assurance and configuration integrity.
Impactful Attacks and Mitigation Measures	Due to the increasing attack landscape, it is necessary to cater for efficient attestation mechanisms to verify software and device integrity for detecting run-time modifications in next generation systems-of-systems. All code-injection, ROP and sophisticated attacks that tamper with state information in the program's data memory (e.g., the stack and the heap) should be able to be detected by the REWIRE attestation schemes (e.g., CFA).
KPIs	• More efficient and scalable attestation techniques compared to the traditional ones due to the reduced ISA space and the monitoring hooks

ID	FR.FR.21 (Mandatory)
Title	**Chain of trust creation**
Actors/Components Involved	TEE, Blockchain infrastructure
Description	**Background:** A major challenge is to guarantee the operational assurance and trust establishment among different and heterogeneous devices in a distributed environment to ensure its trustworthy operation in the context of a collaborative infrastructure. The chain of trust concept is a hierarchical series of trusted components within a system that ensures the security (and the integrity) of its software and hardware components, while part of establishing the chain of trust is the creation of a secure environment where only trusted and verified components may operate and interact with each other. However, restricted and distributed IoT devices does not allow the deployment of more complex protection schemes such as TPMs to support the trust concept. Thus, there is an urgent need for more lightweight software-based solutions that can act as the RoT, the baseline of the trust chain. **Description**: RoT is the cornerstone to facilitate remote attestation protocols by offering the necessary crypto functions that verify the trustworthiness of the underline device. In the context of REWIRE, TEE functionalities will be utilised, as software and more lightweight solution of RoT, to manage the device's keys. More specifically, REWIRE TEE should provide the necessary mechanisms for trust

	management and for creating a chain of trust among heterogeneous and limited IoT devices. In essence, this trust should be verifiable and based on reliable evidence, allowing for transparency and accountability within the IoT ecosystem. This approach will assist on the delegation of critical tasks (e.g., in the automotive domain) in devices proved their trustworthiness. **Remarks**: – Attestation protocols will guaranty the trustworthiness of devices in such a distributed environment.
Technology enablers	Keystone, Hyperledger Fabric,
Connected To Other Requirements	• **FR.FR.17**: **Common Trusted Computing Base**: A common TCB is the baseline as the trust anchor for the creation of chain of trust.
Impactful Attacks and Mitigation Measures	Sophisticated attacks (e.g., code-injection and ROP attacks) should be able to be detected in an efficient manner, minimising the attack surface and increasing the trust among heterogeneous and limited IoT devices.
KPIs	• Scalability of the trust management mechanism among heterogeneous and limited IoT devices. • Time for transmitting the attestation evidence

ID	FR.FR.22 (Mandatory)
Title	**Policy-based device state configuration**
Actors/Components Involved	TEE
Description	**Background**: Complex distributed systems require policies that must be set during design time and implemented during runtime. REWIRE is no exception to this. All use case scenarios will make use of policies for their correct performance. Examples can include (1) the onboarding of a new device, which can include policy-regulated accesses to the network (2) performing SW/FW updates, that must be policy-controlled and attested (3) migrating data and execution to a more secure remote enclave, or (4) revocation of credentials for a suspicious device. **Description**: The REWIRE TEE must express the configuration of the design time phase and must be able to measure itself and report back its configuration. This expression of the design time phase shall allow for policy enforcement and policy-based access to different keys for different uses. Ways to do this must be explored (i.e., will policy enforcement happen at the Security Monitor level? Will some of it run inside enclaves?). Nevertheless, the Security Monitor should be modular to adapt to different devices and different policies. In this regard, the TEE provider could *configure*, build and deploy the security monitor to suit different needs. The REWIRE system requires that different applications in the untrusted world that correspond to basic functions, common to all use cases, communicate either to the Security Monitor or to/from an enclave using the Security Monitor as a means to protect the enclave. This must be performed according to policies provided by the use cases (security models requirements) and using the Security Monitor as the central point for arbitration. The TEE requires that: - The Security Monitor must have the highest privileges, be trustable and have access to the HW root of trust and any crypto co-processor at the HW level, if applicable. Generally speaking, the platform must support storage of a device root key accessible only to the bootloader / Security Monitor. - The Security Monitor must be able to arbitrate communication between the untrusted host and the machine (i.e., allow or deny the protection and execution of an enclave that was initially allocated by the untrusted host). This must be performed following the aforementioned policies, which should cover: o Device Secure Enrolment o Attestation

	○ Function migration ○ Isolation - Beyond arbitration, the Security Monitor must provide keys for securing critical enclave functions such as enclave migration or local and remote attestation. **Remarks:** – The Security Monitor (trusted and highest-privileged component) must express design time phase policies by being able to allow or restrict access of untrusted hosts to the trusted world.
Technology enablers	Keystone
Connected To Other Requirements	• **FR.FR.2**: **Secure remote asset management and reconfiguration effectiveness**: The verification of the SW/FW update provenance determines whether such update can be run inside an isolated memory region. • **FR.FR.3**: **Device status auditing:** Attestation must be policy-regulated • **FR.FR.11**: **Flexible and reliable key management**: The TEE can allow that key generation takes place in enclaves, while some of it can take place in the Security Monitor
Impactful Attacks and Mitigation Measures	Keystone does not natively protect the enclave or the SM against timing side-channel attacks. SM, RT, and eapp programmers should use existing software solutions to mask timing channels. In addition, Keystone hardware manufacturers can supply timing side-channel resistant hardware. Additionally, the RoT protects important assets such as keys of the device (e.g., ownership, attestation, device keys) on shielded or protected locations in the device. The integrity of these locations should be preserved, and they should be prevented from unauthorized modification otherwise the Identity of the device might get compromised and/or the RoT might not be able to verify different pieces of software neither to compute valid signatures.
KPIs	N/A

5.4 Formal verification Requirements

ID	FR.FR.23 (Mandatory)
Title	**Formally Verifiable HW-Security designs**
Actors/Components Involved	The hardware-layer of the REWIRE architecture. Notably, the RISC-V core and an AES co-processor.
Description	**Background**: The hardware design should be formally verified in order to provide guarantees about the required level of trustworthiness when such devices are placed in their operational environments. **Description**: The design phase of REWIRE aims at using formal methods to assert security properties of the proposed hardware architecture, therefore reducing the number of threats that need to be covered/detected by the run-time attestation. Hence, the hardware architecture used in REWIRE needs to enjoy a high-level of trustworthiness regarding its security claims. Examples of such security claims could be protection against side-channel attacks (e.g., Spectre and Meltdown) at the RISC-V core micro-architectural level, for instance. REWIRE should also focus on functional correctness of hardware IPs, resulting in an architecture that is robust against vulnerabilities and exploits stemming from implementation bugs. Regarding the latter, components that are formally verified against a higher-level specification shall be used: one example of such could be a verified functionally correct instance of an AES co-processor. Such AES-co-processor (either tightly or loosely coupled) shall be accessed with a RISC-V custom instruction, which will be part of the REWIRE extended RISC-V ISA. The formal verification tools and languages for achieving this requirement can vary. REWIRE should explore the use of different techniques, including model checking

	(both directly at the model-level and through external tools) and theorem proving (likely with the Coq proof assistant). **Remarks:** – REWIRE should explore the use of different techniques such as model checking and theorem proving.
Technology enablers	Coq proof assistant
Connected To Other Requirements	• **FR.FR.24**: **Expression of requirements and assumptions using formal specification languages**: Expression of requirements and assumptions using formal specification is necessary to formally verifiable HW-security designs.
Impactful Attacks and Mitigation Measures	Micro-architectural side-channel attacks (e.g., Spectre, Meltdown, Implementation exploits / implementation bugs such as buffer or integer over/underflows) can be prevented with formally verifiable HW-security designs. Cache and time partitioning controlled speculative execution and functional correctness proofs are necessary techniques to mitigate these attacks.
KPIs	• Efficiently formally verified HW-Security designs w.r.t the number of ISA instructions.

ID	FR.FR.24 (Mandatory)
Title	**Expression of requirements and assumptions using formal specification languages**
Actors/Components Involved	REWIRE's Security Requirements. The AADL representation of the REWIRE architecture.
Description	**Background**: From the perspective of the design stage, each IoT Service Provider needs to set its overarching requirements with formal descriptions using specification languages. To do so, requirements stemming from the use cases and the REWIRE framework itself need to be expressed using specification languages so that to be given as inputs in the Formal verification framework of REWIRE. **Description**: REWIRE's Security Requirements shall be expressed in formal languages that can be automatically given as inputs for tools such as theorem provers and model checkers. To achieve that, REWIRE shall use AADL plugins for automatic reasoning about requirements. Requirements shall finally be mapped to corresponding Assurance Cases using Resolute, an AADL plugin for formal reasoning. **Remarks:** – By connecting security requirements to a formal compelling argument, trust that every single security requirement is satisfied by the architecture is achieved.
Technology enablers	AADL model
Connected To Other Requirements	• **FR.FR.23**: **Formally Verifiable HW-Security designs**: Expression of requirements and assumptions using formal specification is necessary to formally verifiable HW-security designs.
Impactful Attacks and Mitigation Measures	N/A
KPIs	N/A

ID	FR.FR.25 (Mandatory)
Title	**Provably secure crypto protocols and algorithms**

Actors/Components Involved	REWIRE Design-time architecture, Formal Verification Tools, Provable secure crypto for secure communication, Security models
Description	**Background:** Cryptographic protocols cannot just be based on ad hoc constructions evaluated by attempting attacks against them, as this approach leaves the door open to unexpected attacks or loosely defined security properties. Instead, state-of-the-art protocols must come with precise definitions on the properties they should achieve, and a mathematical proof (reduction) that these properties are indeed enforced based on some well-defined assumptions. In addition, implementations of these protocols should undergo formal verification ensuring that they do meet the protocol's specifications. **Description:** REWIRE will use several cryptographic protocols, including: • A (symmetric) authenticated encryption algorithm, specifically designed for the critical operation of securely distributing SW updates. Depending on the context, this scheme could also be used for transmission of other critical data. • Various symmetric and asymmetric protocols and algorithms allowing operations such as key exchange, digital signature and encryption of regular data exchanges, etc. All exchanges of sensitive data between REWIRE devices should be protected (authenticated and/or encrypted) through cryptographic means. In addition, the authenticated encryption protocol used for SW update distribution should provide protection against side-channel attacks. This authenticated encryption scheme, specifically designed for REWIRE, should be provably secure, and its security proof should take into account the threat of side-channel attacks, to the extent of the current state of the art in this field. Implementations of this scheme designed for REWIRE should be formally verified to ensure it meets its specifications. Other cryptographic protocols used in REWIRE should correspond to widely used good practices, such as industrial or de facto standards. They should ideally be provably secure, unless other constraints (such as interoperability with standards) prevent it. REWIRE will investigate on mathematical and modelling process aiming to provide proofs on the correctness of the crypto implementations and on the security of their primitives. **Remarks:** provably secure schemes will take the form of: – A specification of the scheme, together with a security proof, in the form of an academic paper. – A formal verification of the scheme's implementation
Technology enablers	Keystone
Connected To Other Requirements	• **FR.FR.9**: **Dynamic credential management**: Dynamic credential management is imperative in order to ensure the security of crypto protocols and algorithms. • **FR.FR.11**: **Flexible and reliable key management**: Key management is the baseline to ensure the security of crypto protocols and algorithms.
Impactful Attacks and Mitigation Measures	Several attacks can be prevented when the adopted crypto protocols and algorithms are provable secure such as cryptanalysis and side-channel attacks.
KPIs	• Performance and efficiency of evaluating the security of crypto protocols and algorithms

ID	FR.FR.26 (Mandatory)
Title	**Computer-aided verification of security/safety profiles of systems**
Actors/Components Involved	REWIRE Design-time architecture, Formal Verification Tools, Provable secure crypto for secure communication, Security models

Description	**Background**: The process of designing, implementing, and deploying cryptographic mechanisms is notoriously difficult due to the prevalence of design flaws, implementation bugs, and side-channel vulnerabilities, even in widely deployed mechanisms. Each step of the process is highly complex and filled with potential pitfalls. At the design level, cryptographic mechanisms must meet specific security objectives against a defined class of attackers. This often involves composing intricate building blocks, with abstract constructions making up primitives, primitives forming protocols, and protocols comprising systems. At the implementation level, high-level designs must be translated into concrete functional details such as data formats, session state, and programming interfaces, while prioritizing interoperability and performance. At the deployment level, the implementation must also consider low-level threats, such as side-channel attacks, that were not considered at the design level. All the above strengthens the case for a multi-phase design toolchain supported by automated processes and tools that will support the verification and validation of safety and security profiles for embedded systems. **Description:** A level of automation in the verification processes should be achieved in REWIRE. This may refer only to a smaller part of the verification process. In addition, it depends on the methodologies that will be used (e.g., Theorem proofs or model checking). A toolchain specific to aid the security and safety designs will be based on a mature model-based system engineering approach that will enable both verification of designs as well as, traceability of requirements from systems level (overarching requirements) till low level embedded specifications. **Remarks:** – REWIRE will offer a 4-layered security sandbox comprising a toolchain that can efficiently safeguard and assess the trustworthiness level of an edge device throughout its entire lifecycle and application stack.
Technology enablers	Keystone
Connected To Other Requirements	• **FR.FR.17**: **Common Trusted Computing Base**: A common TCB is the necessary trust anchor for verification of security/safety profiles of systems.
Impactful Attacks and Mitigation Measures	The attack surface should be minimised considering the verification of security/safety profiles of systems.
KPIs	• Efficiently verification of security/safety profiles of systems

5.5 Roots of Trust Capabilities & Properties

Many security and protection mechanisms are currently rooted in software that, along with all underlying components, must be trustworthy [REF-167]. A vulnerability in any of those components could compromise the trustworthiness of the security mechanisms that rely upon those components. Stronger security assurances may be possible by grounding security mechanisms in roots of trust. Roots of trust are highly reliable hardware, firmware, and software components that perform specific, critical security functions. Because roots of trust are inherently trusted, they must be secure by design. As such, many roots of trust are implemented in hardware so that malware cannot tamper with the functions they provide. Roots of trust provide a firm foundation from which to build security and trust.

The hardware root of trust has become the foundation for securing operations in edge computing systems [REF-168] and, as the security system, contains keys for cryptographic functions to enable a secure boot process. The secure implementation of a System-on-a-chip (SoC) design with a root of trust is aimed at protecting the hardware from malware attacks and can act as a standalone security module within the SoC. There are several types of hardware root of trust: one is silicon-based, which falls under both the *fixed-function* and *programmable* categories.

For a fixed-function root of trust, the security module consists of a state machine designed to perform a specific function, such as data encryption, validation, and key management. This type of security module is commonly used in IoT devices. On the other hand, the programmable root of trust is built around a CPU that performs all tasks as a state machine and can also execute a more complex set of security functions.

REWIRE requires openness and interoperability. It also requires ease of integration in different scenarios where security is a cornerstone requisite. Different environments were considered when designing the project, such as Intel SGX and Arm TrustZone, but [REF-169] even though the proprietary TEEs bring many benefits, they have been criticized for lack of transparency, vulnerabilities, and various restrictions. For example, these TEEs only provide a static and fixed hardware TCB, which cannot be customized for different applications. Existing TEEs time-share a processor core with the Rich Execution Environment (REE), making execution less efficient and vulnerable to cache side-channel attacks. Moreover, TrustZone lacks hardware support for multiple TEEs, remote attestation, and memory encryption which are important requirements for REWIRE.

REWIRE's customizable TEE will be based on open-source architectures with a focus on RISC-V to deliver a solution that will ease deployment and system integration, guarantee interoperability on heterogeneous devices, and foster rapid adoption in the community. RISC-V is an open and free ISA, which allows anyone to use, modify, and extend. This customizable TEE will be based on Keystone, an open source full-stack enclave that runs on standard RISC-V cores and which features a modular design for better extensibility & portability.

However, Sanctum was the first enclave design for the RISC-V ISA. In [REF-170], the authors' prototype targeted a Rocket RISC-V core, an open implementation that allowed any researcher to reason about its security properties. Sanctum's extensions could be adapted to other processor cores, since they did not change any major CPU building block. However, they added hardware at the interfaces between generic building blocks (without impacting cycle time). Sanctum's achievements were a foundation for Keystone; indeed, the former demonstrated that strong software isolation was achievable with a surprisingly small set of minimally invasive hardware changes, and a very reasonable overhead. Most of Sanctum's logic was implemented in trusted software, which did not perform cryptographic operations using keys. One of Keystone's pillars, PMP, was however introduced in 2017 (RISC-V Priv. v1.10), rendering Sanctum's non-standard hardware extensions obsolete.

Before we concentrate on the root of trust, let us consider that a Keystone-capable system consists of several components in different privilege modes [REF-171]:

Trusted Hardware. A CPU package built by a trustworthy vendor, which must contain Keystone-compatible standard RISC-V cores and root of trust. The hardware may also contain optional features such as cache partitioning, memory encryption, cryptographically secure source of randomness, etc. The Security Monitor requires platform specific plug-ins for optional feature support.

Security Monitor (SM). It is M-mode software with small TCB. The SM provides an interface for managing the lifecycle of enclave as well as for utilizing platform-specific features. The SM enforces most of Keystone's security guarantees since it manages the isolation boundary between the enclaves and the untrusted OS.

Enclaves. Environments isolated from the untrusted OS and other enclaves.

Enclave Applications (eapp). An eapp is the user-level application that executes in the enclave.

Runtime is S-mode software which implements functionality such as system calls, trap handling, virtual memory management and so on for the eapps.

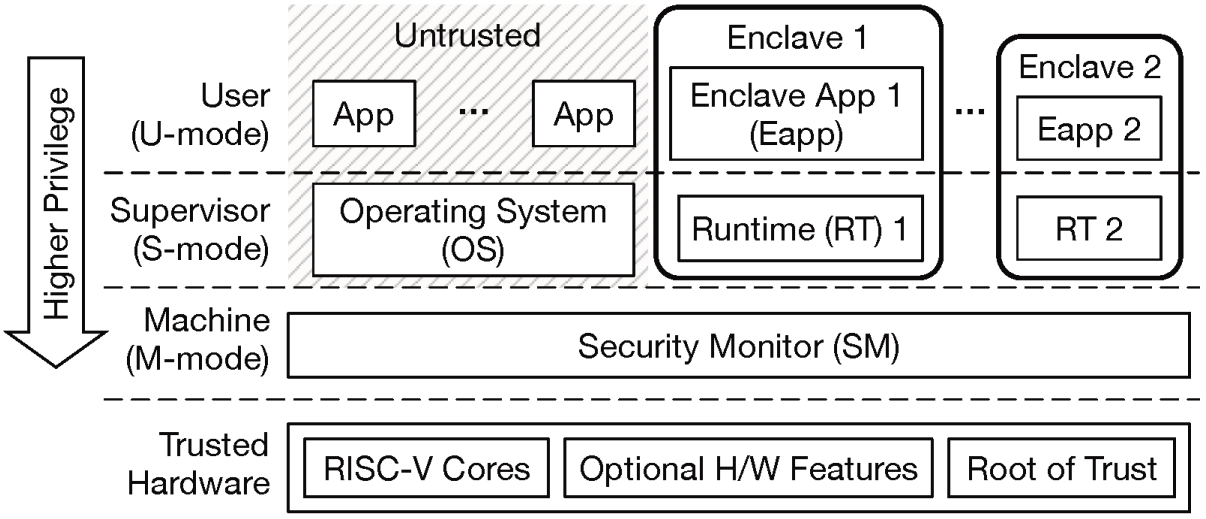

Figure 5.1: A Keystone-capable system and its components

Keystone root-of-trust can be [REF-172] either a tamper-proof software (e.g., a zeroth-order bootloader) or hardware (e.g., crypto engine). At each CPU reset, the root-of-trust (a) measures the SM image, (b) generates a fresh attestation key from a secure source of randomness, (c) stores it to the SM private memory, and (d) signs the measurement and the public key with a hardware-visible secret. These standard operations can be implemented in many ways. Keystone does not rely on a specific implementation, Lee et al. [REF-173] prefer a crypto engine, nonetheless. The basis is in this case for the hardware to have a unique and immutable cryptographic identity. Besides authenticity, it is mandatory that the processor demonstrates that it has booted correctly and for this it can present a container measurement certificate.

Different scenarios can take place for a system to accomplish this secure boot: In one setting [REF-174], the processor hardware cryptographically measures the bootloader software and all software loaded by the bootloader, then uses the processor's *secret* key to sign the measurement, producing a certificate. The public key associated with the processor's secret key is signed by the manufacturer. The local or remote client verifies the signature and measurement using the processor's public key. To produce such a certificate the processor must be provisioned with a public key pair. Generally, the manufacturer produces a key pair for each chip and embeds the private key into the chip, typically into secure non-volatile memory. In this scenario however, the manufacturer knows the chip's private key without the chip leaking it so even a correctly manufactured processor could have exposed keys at production.

In an alternative setting, to maintain *privacy of the secret key*, a hardware True Random Number Generator (TRNG) can generate a random seed stored in secure non-volatile memory, which is used to generate a public/private key pair inside the processor. The manufacturer signs the public key, which can be stored in publicly readable non-volatile memory while the private key never leaves the processor and is unknown to the manufacturer.

Yet a more recent approach is the use of silicon Physical Unclonable Functions (PUF), which extract volatile secret keys using semiconductor manufacturing variations that manifest when the chip is powered. PUFs are used as *symmetric key* generators in different commercial products. A PUF can be used to generate a random seed for a public/private key generator inside of a secure processor but there are no known published implementations of this.

Lebedev et al. [REF-175] used a RISC-V Rocket chip architecture as a base to implement an attested execution processor which derived its cryptographic identity from manufacturing variations measured by a PUF. In this case, a bootloader built into the processor transformed the PUF output into an elliptic curve key pair, the public portion of which is signed by the manufacturer using its private key.

All REWIRE components that at some point interact with the TEE require that – firstly – such trusted

environment has booted in a verifiable state, which means that every step in the boot process can be and is attested to the client. The bootloader is considered then the root of trust for the boot process.

Different strategies can be taken at this point, and REWIRE's root of trust requirements must dictate the ones to choose as to:

- How our secure boot process will obtain its attestation root key
- Whether the root of trust for measurement/verification differs from the root of trust for attestation
- Whether components are verified using a signature from the manufacturer or by a measurement of their code

In conclusion, REWIRE's TEE implementation will be agnostic to the underline RoT, even though the implementation will be based on the RISC-V open-source architecture. On top of that, for the actual customizable TEE, the open-source Keystone will be used due to its modular design, which is a highly desirable feature for REWIRE.

5.6 RoT Requirements for REWIRE

ID	FR.FR.27 (Mandatory)
Title	Trusted Root-of-Trust and secure boot
Actors/Components Involved	TEE
Description	**Background:** Many of REWIRE's data and applications rely on the use of TEEs in the ecosystem's devices / components, but for the former to be trustworthy they need to incorporate a secure-by-design trusted RoT. The RoT, for all intents and purposes, must be (as) inaccessible (as possible) outside the device, and in this way the device can trust the keys and other cryptographic information it receives from the RoT module and other components in the ecosystem can trust the device. The RoT must remain the foundation on which other secure operations depend and thus must contain *protected* keys for cryptographic functions to enable a secure bootstrapping process – this including a secure SM boot process. Every device / component in the REWIRE ecosystem must be able to boot using exclusively trusted and authenticated firmware / software. Otherwise, the device might reboot at some point and run malicious code, rendering the device compromised. **Description:** The RoT (e.g., a hardware based crypto engine) must cryptographically measure the bootloader software and all software loaded by the bootloader, and then use the processor's private key to sign the measurement, producing a certificate. The public key associated with the processor's private key is signed by the manufacturer and it is known by others. The privacy of the secret key must be guaranteed (several methods can be explored, like using TRNG or PUFs). As part of the fully secure boot process, the RoT must also authenticate the SM. Specifically, at each CPU reset, the RoT must measure the SM image, then generate a fresh attestation key from a secure source of randomness, store it to the SM private memory, and signs the measurement and the public key with the hardware-visible secret key mentioned earlier. As seen, the RoT protects important assets such as keys of the device (which are then used for other security-critical functions like attestation) on shielded or protected locations in the device. The integrity of these locations should be preserved, and they should be prevented from unauthorized modification otherwise the identity of the device might get compromised and/or the RoT might not be able to verify different pieces of software (e.g., the SM) or compute valid signatures. During the operation of the device, the SM can provide security services to the TEEs or to other applications. If these services are anchored on the RoT, the SM should be able to attribute the services to the owning entity. **Remarks:** - A secure-by-design RoT (e.g., a hardware based crypto engine) must be able to

	cryptographically measure the bootloader software, the Security Monitor and any other software at boot-time. - The TEE RoT in REWIRE can act as the foundation for critical security services / concepts such as ownership, integrity and authenticity. - (Relations of trust) The Keystone user trusts the SM only after verifying if the SM measurement is correct, signed by trusted hardware (RoT), and has the expected version. The SM only trusts the hardware, and the host trusts the SM.
Technology enablers	Keystone
Connected To Other Requirements	• **FR.FR.3**: **Device status auditing**: Secure measurements and attribute extraction are necessary for auditing of the state of the device. • **FR.FR.17**: **Common Trusted Computing Base**: At the heart of the TCB we have the underline RoT. • **FR.FR.20**: **Operational assurance and configuration integrity**: The first step to assure correct operation and configuration integrity is the trusted boot. • **FR.FR.21**: **Chain of trust creation**: The main idea behind the chain of trust is to have a trusted RoT. • **FR.FR.31**: **Platform Measurement Service**: In order for a TCB to provide valid measurements should be trusted.
Impactful Attacks and Mitigation Measures	A physical attacker can intercept, modify, or replay signals that leave the chip package. However, we assume that the physical attacker does not affect the components inside the chip package. Also, keystone does not natively protect the enclave or the SM against timing side-channel attacks. SM, Runtime, and eapp programmers should use existing software solutions to mask timing channels. In addition, Keystone hardware manufacturers can supply timing side-channel resistant hardware.
KPIs	• Elapsed time to complete a trusted boot in a device.

ID	FR.FR.28 (Mandatory/Optional)
Title	**Bootstrapped RoT**
Actors/Components Involved	TEE
Description	**Background:** GlobalPlatform, depending on the RoT implementation categorizes the RoT to either bootstrapped or a non-bootstrapped RoT [10]. Today, several designers of security protocols, schemes, products, and services rely on a bootstrapped RoT to provide security. In a nutshell, a bootstrapped RoT is composed of one initial RoT and one or more extended RoT components, which together make up a bootstrapped RoT. **Description:** The TCB that will be used in REWIRE includes both a HW RoT and the SM (e.g., Keystone). The SM will be in charge of managing the HW and/or a cryptographic engine and will guarantee that the TEEs have certain security properties (e.g., isolation or integrity). The SM must be modular to adapt to the different devices and their needs. Furthermore, REWIRE might offer the option to provide additional features. Therefore, REWIRE considers a combination of an Initial RoT Component and the extension with more RoT Components [10]. In the end, after the booting process, it should be possible to validate different versions or flavours of the SM. **Remarks:** • REWIRE's bootstrapped RoT includes both a hardware and software RoT (e.g., Keystone's SM). • REWIRE's customizable TEE is based on RISC-V to guarantee hardware- and software-trust on IoT devices due to the ability to use, modify, and extend.
Technology enablers	Keystone
Connected To Other Requirements	• **FR.FR.17**: **Common Trusted Computing Base**: The extensions of a bootstrapped RoT will be used to provide a common TCB. • **FR.FR.21**: **Chain of trust creation**: Chain of trust will be achieved with a bootstrapped RoT that includes both a hardware and software RoT.

Impactful Attacks and Mitigation Measures	REWIRE minimizes the attack surface since the components of the bootstrapped RoT are trusted inherently.
KPIs	• Light footprint w.r.t power consumption (e.g., RoT with the minimum function list and low complexity that still supports all of the required functions)

ID	FR.FR.29 (Mandatory/Optional)
Title	**Immunity to Software Attacks and Integrity protection**
Actors/Components Involved	TEE, RoT
Description	**Background:** The increasing number of online IoT and edge devices exposes them to several software attacks. RoT is the baseline to provide immunity to these remote software attacks and assuring the integrity of the device. Ideally, immunity to software attacks and integrity protection should also protect against physical attacks, however in reality physical attack protection is difficult. Software RoT is mainly designed to resist software-based attacks, however if correctly designed it can also protect from hardware-based attacks. **Description:** As the components of the RoT are inherently trusted and are supposed to be secure by design, it should be guaranteed that the security critical functions that they perform cannot be compromised via software attacks. These functions can be implemented in hardware and/or protected firmware, but in any case, it should be guaranteed that they cannot be modified via a software attack. Additionally, the RoT protects important assets such as keys of the device (e.g., ownership, attestation, device keys) on shielded or protected locations in the device. The integrity of these locations should be preserved, and they should be prevented from unauthorized modification otherwise the Identity of the device might get compromised and/or the RoT might not be able to verify different pieces of software neither to compute valid signatures. **Remarks:** – REWIRE's customizable TEE as the RoT should provide security form software attacks and integrity protection to all the critical functions.
Technology enablers	Keystone
Connected To Other Requirements	• **FR.FR.27**: **Trusted Root-of-Trust and secure boot**: A trusted RoT will be in position to provide immunity to software attacks and integrity protection. • **FR.FR.30**: **Identifiable Ownership**: The RoT should prove the ownership validity thus, is necessary to be immune to software attacks and have integrity protection. • **FR.FR.31**: **Platform Measurement Service**: The RoT should provide an assurance that that measurements are correctly captured thus is necessary to be immune to software attacks and have integrity protection.
Impactful Attacks and Mitigation Measures	REWIRE's customizable TEE should be immune to software attacks. To prevent side-channel attacks from succeeding, the RoT needs to have built-in resistance. Different types of attacks (invasive or non-invasive) require different counter measures. It is also crucial to find the trade-off between the protection and the penalty such as power consumption.
KPIs	• Percentage of software-based attacks that can be successfully identified

ID	FR.FR.30 (Mandatory/Optional)
Title	**Identifiable Ownership**
Actors/Components Involved	RoT, TEE
Description	**Background:** The control and ownership of the RoT is of paramount importance. The environment around the RoT evolves as time goes by, thus it is crucial for authorized users to be able to attribute the services provided by the underline RoT (e.g.,

	REWIRE's TEE) to the owning entity. Especially in cases where the ownership of the RoT is different from the ownership of the platform containing the RoT. **Description:** During the operation of the device, the SM can provide security services to the TEEs or to any other application. If these services are anchored on the RoT, the SM should be able to attribute the services to the owning entity. Furthermore, If the code of the RoT must be updated due to the discovery of a critical bug, it should be guaranteed that only the owner of the RoT can perform such update. Note that the owner is already trusted and that is one way to maintain the trust after the update. **Remarks:** • A RoT shall have a single identifiable owning entity.
Technology enablers	Keystone
Connected To Other Requirements	• **FR.FR.29**: **Immunity to Software Attacks and Integrity protection**: The RoT should have immunity to software attacks and integrity protection to prove that the ownership is valid.
Impactful Attacks and Mitigation Measures	REWIRE's TEE should have immunity to software attacks and provide integrity protection. Impersonation if the owner key is leaked. As mitigation provide a re-keying mechanism or a way to transfer ownership
KPIs	N/A

ID	FR.FR.31 (Mandatory/Optional)
Title	**Platform Measurement Service**
Actors/Components Involved	RoT, TEE
Description	**Background:** Endpoint devices contain hardware, firmware, drivers, OS, and software that affect the integrity and security of the devices and the network within which they reside. Thus, a service providing knowledge on the current posture of these endpoint devices is of paramount importance. Platform integrity measurement is a core service of attestation mechanisms based on which can be decided whether the platform is in a correct state or not. Platform integrity measurement is important for building up a trusted environment. This service provides the ability to reliably create characteristics of the platform by calculating hashes of core and/or data. Current measurement methods suffer from high computational complexity and heavy data processing, thus they consume a lot of resources and spending too much time. **Description:** One of the security services that the SM offers is a measurement service of the platform characteristics, in such a way platform users can rely on a report that describes the current state of the platform and is emitted by this platform and signed by the SM and/or by the device identifier. The SM can prove to a remote client that some enclave contains the program expected, and is running on hardware that is trusted. This report might be a cryptographic hash with a list of features or services provided by the RoT. **Remarks:** • Integrity measurements may be in the form of cryptographic hash. • The underline RoT must act in concert to enable reliable and trustworthy measurements.
Technology enablers	Keystone
Connected To Other Requirements	• **FR.FR.27**: **Trusted Root-of-Trust and secure boot**: A trusted RoT will be in position to provide valid platform measurement. • **FR.FR.29**: **Immunity to Software Attacks and Integrity protection**: The RoT should have immunity to software attacks and integrity protection to provide a valid platform measurement.

Impactful Attacks and Mitigation Measures	REWIRE's TEE should have immunity to software attacks and provide integrity protection.
KPIs	• Time elapsed for the actual measurement. • Light footprint w.r.t power consumption for the actual measurement

Chapter 6
REWIRE Use Cases

This chapter describes the three REWIRE use cases along with their corresponding user stories, the use case requirements, and the metrics of success (KPIs). This step is necessary in order to define the REWIRE MVP.

6.1 High Level Introduction of the REWIRE Use Cases

Throughout the project's duration, three distinct Use Cases will be put into action. These Use Cases will serve as instruments for assessing the effectiveness of the REWIRE approach across various vertical domains. Their deployment will not only offer insights into the overarching implementation but will also contribute to the articulation of functional requisites and domain-specific demands that the REWIRE technology features must address (see Section 6). In this context, the comprehensive REWIRE design will be coupled with the establishment of suitable cryptographic trust anchors within the realms of hardware and software-based Roots-of-Trust, along with the deployment of efficient cryptographic primitives. This, in turn, ensures the secure and trustworthy operation of digital trust throughout the entire lifespan of the devices. The objective is to validate the foundational principles of the REWIRE framework, encompassing remote attestation (along with the underlying trusted computing technologies), lightweight cryptography, real-time risk assessment, and the secure, auditable sharing of operational threat intelligence data flows. These facets are incorporated with policy compliant Blockchain infrastructures within the envisioned REWIRE industrial Use Cases. In this context, the Use Cases serve as a valuable source of input for shaping the architectural framework of the overall REWIRE platform (referenced in D3.1, D4.1, and D5.1). Additionally, the Use Cases serve as the cornerstone for evaluating not only the framework itself but also the demonstration scenarios extracted in WP6.

The various use cases will be implemented through a series of scenarios and user stories, as detailed in Section 5. These scenarios require the instantiation of the REWIRE platform to function as intended. It's important to note that the user stories associated with each use case may not explicitly showcase all the underlying REWIRE key technologies and services. In many instances, these technologies operate in the background, imperceptible to end users. Nevertheless, the successful execution of these user stories and the seamless operation of each scenario hinge on the effective integration of the methods provided by REWIRE into the use cases.

Consequently, each user-story scenario will be complemented by a user-story confirmation and an initial mapping to the REWIRE technology features, with confirmation of validity contingent on the completion of technology requirements for each feature. This preliminary mapping represents the essential REWIRE security and privacy-preserving services required to address the critical challenges identified in each targeted application domain within REWIRE. These mappings may undergo further refinement in D6.1, where the specific evaluation and demonstration plan for each use case will be compiled.

For the initial alignment of each use-case scenario with the technical prerequisites of REWIRE, please refer to Section 6. It will serve as a guide, clarifying how the technology readiness for each user-story will be gauged and advanced over the project's duration.

Table 6.1: High Level Introduction REWIRE Use Cases

Use-case	Use-case Title	Focus on REWIRE feature	Responsible Partner
1	Smart Cities - Smart Cities for Empowering Public Safety	Secure software distribution, secure onboarding, verifiable attestation, runtime analysis, AI-based threat detection, attribute-based access	ODINS

		control	
2	Automotive - Adaptive In-Vehicle SW & FW Patch Management & Software Functions Migration	Secure software distribution, secure onboarding, verifiable attestation	KENOTOM
3	Smart Satellites - Smart Satellites Secure SW Updates for Spacecraft Applications & Services	Secure software distribution, secure enrolment, secure onboarding, verifiable attestation, AI-based misbehaviour detection	LSF

In the sections that follow, each use-case will delineate its reference scenarios, which comprise user-stories facilitating the shift from current ("as-is") practices to future, secure "to-be" practices and deployments, leveraging the REWIRE technology. It is essential to emphasize that the use-case scenarios align closely with the distinctive features of the REWIRE technology. Their specific goals are contingent on the individual business models pursued by each industrial partner. Consequently, the validation of user-stories will confirm the realization of each user-story, made possible through the capabilities of REWIRE.

6.2 Use Case – 1: Smart Cities for Empowering Public Safety

IoT has shown to be able to offer improved solutions for the modernization of Smart Cities deployments. Recently, wireless sensor networks have made it possible for autonomous wireless sensors and actuators to monitor city infrastructures while covering enormous regions with fewer base stations. Besides, some of these IoT networks provide access in remote areas without cellular coverage (4G or 5G). The high expectations placed on the IoT paradigm explain this exacerbated increase in device density, but is this paradigm mature enough to bear the responsibility that will fall on it? A look at the current landscape shows that there is still room for improvement.

To avoid hindering the advancement of technology and industry, the proliferation of vendor-specific ecosystems without compatibility with the Internet must be avoided. Standard defining organizations --- such as the IETF or the IEEE) --- are making efforts to encourage the adoption of new working methodologies in which common communication protocols and data formats become relevant. This is done with compatibility in mind, laying the foundations to facilitate the scalability of the IoT network.

The evolution of this sector is inherent to the adoption of standardization, but there is another aspect that, if not put into perspective, can be neglected: security. The limitations of current technology may hinder IoT security, but this is no longer an excuse to relegate it to the background. Since IoT will become a critical element of human infrastructure in the future (e.g., Intelligent Transportation Systems, Smart Agriculture, Industry 4.0), we must ensure that these assets are secure against threats. Any vulnerability introduced in today's planning will expose the global production capacity to crippling blows in the future.

In the upcoming Smart City scenarios, several challenges must be faced by state-of-the-art IoT solutions and deployments. Compatibility of different vendors' heterogeneous IoT systems and devices for example. The usage of several protocols and mechanisms by existing devices, processing units, and Decision Support Software (DSS) platforms makes interactions among them extremely challenging. Open standardized protocols are essential for enabling interoperability between components and deployments using technologies from various vendors. New networked and interoperable ecosystems are replacing isolated solutions with vendor dependence in Smart Cities. Furthermore, due to the distributed and physically dispersed architecture created by the components and services from various suppliers, they are vulnerable to security and privacy risks. Numerous large IoT device deployments and networks have increased the number of security attack vectors available. Attackers may take advantage of numerous devices which operate for long periods of time without supervision. This specifically impacts administrators

who deploy IoT devices without specialized knowledge or tools. In order to strengthen next-generation IoT services for the lucrative commercial area of Smart Cities, standardized protocols are essential to securely perform activities such as authenticated key exchange and end-to-end object encryption.

6.2.1. "As-is" Scenario

Figure 6.1 showcases the common scenario architecture for IoT sensor networks in the context of Smart Cities. End-devices are IoT sensors and actuators located on-site, gathering information, and providing actuation features on their physical environment. These devices communicate directly through a myriad of radio communication technologies, with edge network infrastructure components — e.g., WiFi routers or cellular network base stations. These, in turn, forward the collected data through a backhaul network — typically the Internet — towards a set of centralized servers which run the high-level data-processing procedures and maintain the overall health of the network deployment. Finally, data is gathered by several cloud services or applications, which pull the information stored in the centralized servers.

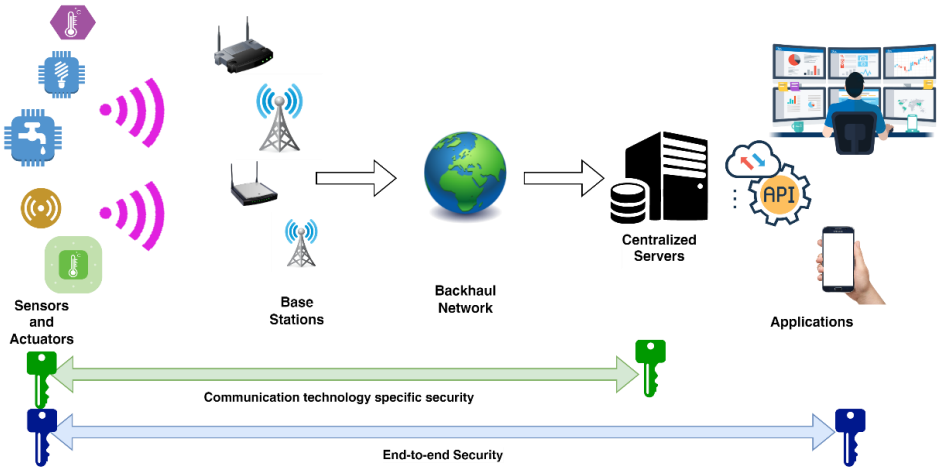

Figure 6.1: IoT Smart City communication architecture

Smart City services focus on providing services to the citizens capitalizing on the IoT paradigm as well as wireless sensor networks and other ICT innovations running on the cloud. The services that are more relevant for a City Hall are those that improve the safety and wellbeing of their citizens. For this reason, when applying ICT to form these services, cybersecurity becomes a critical factor due to the potential personal or material losses incurred in the case of a successful cyberattack or misconfiguration of the system. For these reasons, in this use case of REWIRE, the most relevant public safety services provided in the context of Smart Cities are studied — (i) fire detection and suppression systems, and (ii) vehicle and pedestrian traffic control systems.

In a fire detection and suppression system, end-devices are connected to several temperature and air sensors that continuously monitor their physical environment in order to raise an alarm when fire or smoke is detected. In this event, a signal is sent to a centralized platform — typically managed by the Firefighting services — that is warned of the event. In some instances, additional fire suppression measures can be activated on site, such as audible and light alarms or fire sprinklers.

In vehicular and pedestrian traffic systems, some facilities are put on-site in order to improve traffic efficiency without risking the safety of citizens. For instance, in a pedestrian crossing with traffic lights, a button can be pressed to request crossing. Typically, the programmable logic controllers installed in cities manage several different traffic lights and crossings nearby. These controllers do not have a fixed logic that runs 24h but may support features such as scheduling events that prioritize traffic in certain directions or reacting to external events, such as vehicle detectors through magnetic spires under the asphalt.

In both aforementioned cases, devices are deployed without human supervision, and can be manipulated or maliciously tampered. If the tampering goes undetected, devices with network connectivity can be

accessed remotely by the attackers at a later date, going unnoticed by the network administrators.

Regarding security, most deployments rely on both vendor specific security technology, and end-to-end security mechanisms at application level. Depending on the technical specifications of the end-devices and their quality requirements, the devices can support only certain security protocols. Mostly, the selection is predetermined by the computational capabilities of the device, as well as the aggregated network bandwidth available — larger networks concentrate on saving bandwidth over QoS for scalability purposes.

However, despite the adoption of such security measures, the attack surface of IoT devices remains wide, leaving room for attackers to compromise critical IoT infrastructures. We need to consider that IoT devices nowadays are critical components that may affect safety-critical environments. Thus, it is of paramount importance to seek solutions that will minimize the attack surface of IoTs and push forward the security and safety by-design concept.

Nowadays, the infrastructure usually consists of ad-hoc solutions built around legacy devices that have been given connectivity through the installation of IoT gateways. In rare cases deployments are mature enough to be based on standardized technologies as it is common to find infrastructure based on proprietary technology, which hinders seamless connectivity. This produces systems with very large attack surfaces due to the large number of devices and impracticality of installing newer ones.

Device onboarding in Smart Cities ecosystems

Onboarding is a process in which new users or devices are added to a system and given access to its features and resources. In the context of authentication and authorization, onboarding refers to the process of verifying the identity of new users and granting them access to the system's resources based on their level of authorization.

During the onboarding process, devices are typically required to provide some form of authentication, such as a username and password, or more advanced methods such as public keys, to prove their identity. Once their identity is verified, the device is assigned an appropriate level of authorization based on their role or permissions within the system. Thus, based on its role on the network an on–boarded device may be given access to higher or lower privilege resources.

Within the context of Smart Cities, the onboarding typically happens whenever the device is deployed on-location the first time, or when the device ownership switches administrative domains — e.g., when a company merger happens, the new company policies may require that the devices obtain the secure key from a different endpoint. Additionally, installation technicians have to often deploy large amounts of devices in one sitting to save costs, thus, onboarding processes are expected to be scalable and support up to hundreds or thousands of devices deployed in the same day. Overall, device onboarding is a tedious process which is performed manually with low level of automation.

Figure 6.2 shows the typical onboarding scenario, which is split in two different stages. First, during the Authenticated Key Agreement (AKA), devices exchange messages with an authentication server over a non-secure channel, which is often done using lightweight crypto — e.g., AES-128. Once the procedure is finished, both endpoints derive a shared symmetric session key. Then, the onboarding is considered finalized. Next, during the regular life-cycle operation of the device, both ends employ the obtained key to protect data.

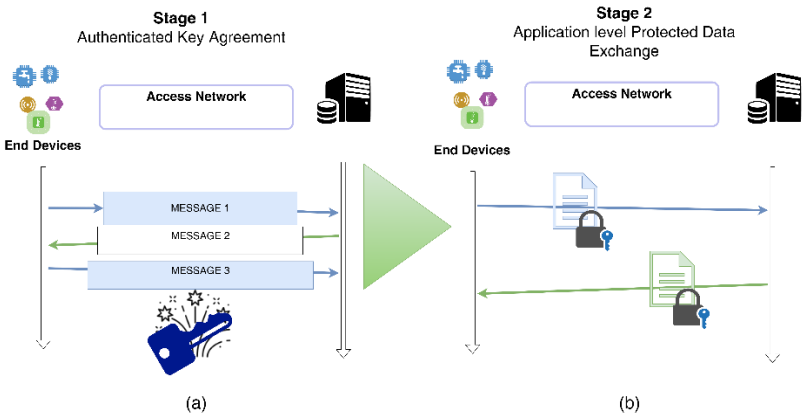

Figure 6.2: Smart City Secure Device On-boarding

Misbehaviour Detection in Smart Cities

Collaborative Threat and Misbehaviour Detection refers to a technique for detecting and mitigating security threats and undesirable behaviour in distributed systems, such as online social networks, cloud computing systems, or peer-to-peer networks. The approach involves leveraging the collective intelligence of a community of users to identify and address potential security threats and misbehaviour. IoT deployments still count on expert users appointed to monitor the overall status of networks. These engineers trust their knowledge about device behaviour, as well as their domain on network flows. Unluckily, while somewhat reliable, this approach is not scalable and lacks objective criteria that ensure the correct recognition of anomalies.

During the design stage of the application and business vertical logic, the development and data flows of IoT Smart City devices are relatively predictable. Hence, developers know in advance the expected communication and behaviour patterns that devices must follow. By employing this information, systems can more effectively detect and respond to security incidents that may be difficult to identify through traditional security measures.

Over the air updates and device reconfiguration in Smart Cities

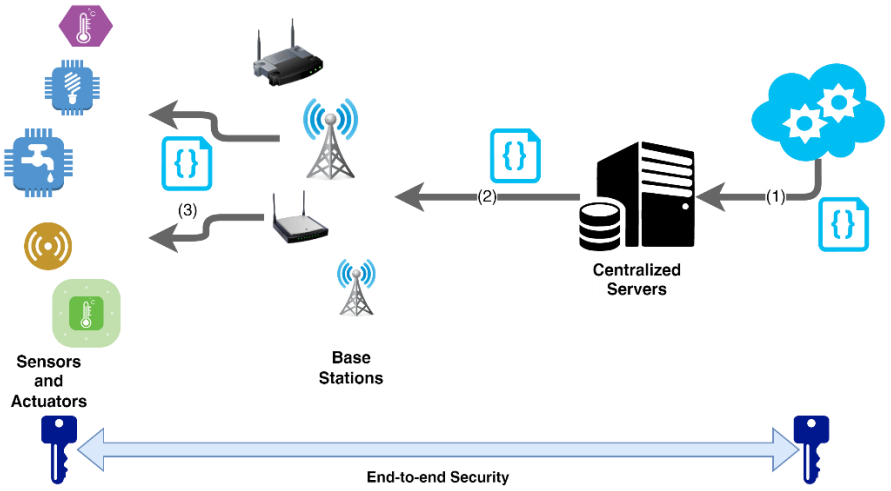

Figure 6.3: Smart City FUOTA scenario

A typical Smart City firmware update over the air (FUOTA) scenario is depicted in Figure 6.3. Whenever a new version of the firmware for end-devices is ready to be rolled-out, first, a purpose-specific service, with the appropriate authorization, sends the new firmware version to the centralized server through external APIs (1), indicating to which devices this must be distributed. Next, the Centralized servers of the Smart City deployment trace a new route for the firmware to be transmitted and it is sent through the

backhaul network towards the radio edge communication components (2). Finally, the firmware is transmitted to the end-devices through the employed radio access technology (3). The end-device checks and validates the firmware using end-to-end security mechanisms.

Currently, commercial solutions seldom provide any sort of verification or attestation that the FUOTA procedure has proceeded securely and successfully. Typically, after an update, the device will not send warnings or acknowledgements that the FUOTA took place, and it is implicit when receiving the expected business logic application data that the update took place.

6.2.2. Scenario's Challenges and Needs from REWIRE

Given the analysis of the previous section on the "as-is" state of the smart cities' ecosystem, we highlight below the challenges and needs that should steer the REWIRE developments.

- **Validation of updates:** End-devices are installed on-site without regular human supervision. When performing updates over the air, there are no current mechanisms that protect the device against physical attacks or report procedure status to the administrators. This means that the firmware update procedure is a particularly sensitive moment in the device's life cycle. In addition, there are no guarantees that the firmware which is being deployed is free of vulnerabilities. Thus, over the air updates on multiple devices may result to a smart city environment with a wide attack surface.
- **Lack of authentication of the SW/FW update origin**: End-devices commonly trust the administrative authority of the components connected to their network domain. Thus, when a signal is received to start the firmware update procedure, no further checks are performed, and validity is assumed by the device. This means that unauthorized users can trigger firmware updates without the consent of the legitimate network administrator and program malicious firmware. Thus, authenticated and secure communication is needed to guarantee the delivery of the updates.
- **Post-management of updates:** Within the Smart-Cities Scenarios, in the Critical Infrastructure subset (.i.e., IoT systems managing infrastructure considered essential for the safety of the population) it is important to ensure the integrity of the entire deployment, therefore it is not acceptable that during an update some sensors unexpectedly malfunction — e.g., a traffic light presenting an unexpected behaviour after an unsuccessful update will most probably cause an accident. This is a problem as the FUOTA mechanism do not have a status report procedure to inform the administrators. Therefore, it is needed to ensure that the updates reach their destinations without any data-loss, log any update-related events, and monitor the stability of the devices after they have been updated.
- **Threat detection based on devices' behaviour analysis**: While sensor-related technology continuously improves over the years, due to their electro-mechanic nature, it is unavoidable to experience some casual mishappens. E.g., a faulty temperature sensor might report false readings, or due to being accidentally hit by pedestrians, wildlife, or vegetation. In those instances, measurements typically produce outlier values or trigger excessive bandwidth due to event-driven measurements (e.g., physical parameter over certain threshold). Security experts monitoring the deployment might interpret these anomalies as a threat and raise a false alarm. A combination of these false alarms might impose an administrative overhead on the deployment owners. Mechanisms to tell apart device faults, actual hazards, or malicious data corruption are highly needed.
- **Device integrity assurance:** Devices are located in remote places and are unattended, making it difficult to ensure the system integrity and operational assurance. Therefore, there is a need for robust security mechanisms that can infer the correct status of the device to guarantee its integrity, especially when these are destined to operate in the context of safety critical services.
- **Traffic analysis capabilities for malicious payload:** Even if we ensure that updates are launched from legitimate agents in our network, there is still a risk that malicious entities gain access to one of the SW/FW distributor nodes and adjust their behaviour to distribute malware. To prevent compromised devices from infecting the network, it is necessary to apply authentication

and integrity mechanisms on traffic payloads.

- **Zero Trust Onboarding:** Due to the large scale of IoT deployments, it is common to find administrative domains that manage hundreds or even thousands of devices in a single radio cell. Additionally, these devices have a typical life cycle of several years of operation. During that period, it is possible that the network administrators install different network infrastructure elements — e.g., to improve QoS, or because devices face power blackouts or intentionalfallback deactivation for maintenance. Hence, the devices will attempt to connect to the network through this newly installed or recently discovered equipment. For this reason, devices are considered as nomadic — i.e., although they are not mobile in essence, occasionally they may connect to other infrastructure network components. This presents a challenge since devices may come and go into the deployed network. Current market solutions relapse into (re)joining the network in case of these events, however, during this (re)join procedure, no further security concerns are taken into account — this increases the risk for attacks.
- **Strict onboarding access:** Authorization and authentication are performed regardless of the state of devices. Currently compromised devices may be accepted in the network. There are no checks on the system integrity prior to the onboarding of devices in the network. This lack of security checks in the onboarding process can result in compromised devices being allowed into the network, which can pose a significant security risk.
- **Protection from data extraction**: Most market solutions envision their end-devices in an install-and-forget fashion. This means that the devices are installed in vast geographical areas by the technician and are not supervised by humans. For this reason, attackers may get hold of some of these devices and extract data. Therefore, tampering with data might result in leakage of crypto material stored in the device.
- **Seamlessly integrable:** It is worthy of note that the most relevant criteria for the successful implementation of additional security enhancements in any deployment is the seamless operation of their devices. As a consequence, the regular business logic of the sensors must not be altered by security procedures. Faulty security-related procedures might interfere with the regular operation logic (e.g., not taking into consideration the device calibration or accuracy and reliability of data).

6.2.3. “To-be” Scenario

Given the challenges analysed in the previous section, the smart cities REWIRE pilot for public safety aspires to demonstrate and address the aforementioned challenges in the following ways:

- **Validation of updates:** Ensuring the security and integrity of the firmware update process to prevent unauthorized access or tampering. The FUOTA will be enhanced by the addition of SW/FW validation processes based on static and dynamic analysis techniques to guarantee that the SW/FW is free of vulnerabilities and implementation flaws. This is needed as the current status implies that no validation of the SW/FW update takes place prior to the update process.
- **Authentication of the SW/FW update origin**: Secure communication channels and authentication mechanisms will be employed to ensure that only authorized users can perform firmware updates. The FUOTA will be enhanced by the REWIRE side-channel resistant authenticated encryption schemes in order to guarantee the security and authenticity of the transmitted update. In this way, any update procedure started by a component connected to the device network must be authorized only if in hold of the needed credentials and crypto to ensure the integrity and authenticity of the update.
- **Post-management of updates**: Feedback is vital to ensure the success of widely applicated updates and REWIRE addresses this through a continuous monitoring after FUOTA events in order to assess end-device’s runtime. The output of these operations will

be used to produce claims which will be attested to provide verifiable evidence of end-devices status.

- **Threat detection based on devices' behaviour analysis**: Detect any outlying values that could affect the operational behaviour of devices or the network activity through an AI-assisted misbehaviour detection mechanism, which will receive as an input network/system data and will intelligently detect the potential security threats.
- **Device integrity assurance:** Devices must be disposed with the proper attestation schemes to provide verifiable claims about the status of its operations. Thanks to these security claims (stored on an on-chain storage) REWIRE components will have a verifiable data source to keep track of device's integrity — I.e., correct bootup, onboarding, successful updates, etc — or detected potential vulnerabilities and attempts to compromise system integrity.
- **Data payload monitoring:** In order to differentiate anomalous but legitimate readings from readings that have been produced by faulty sensors/devices, it must be possible to directly analyse packet's payload in search for patterns typical of malfunctions or attacks. Captured traffic through Secure Oracles Input will be provided through BC infrastructure as an input for the continuous training of the AI-assisted misbehaviour detection mechanisms and as reference data when requested to perform analysis operations.
- **Traffic analysis capabilities for malicious payload:** Traffic transmitted over the network when performing updates must have mechanisms to confirm the legitimacy of the SW/FW regardless of the reputation of the source node. This is achieved through the use of mechanisms such as the TEEs and the data filtering of the Secure Oracles.
- **Strict onboarding access:** The architecture of the system must dispose of the mechanisms so the access to the onboarding process is strictly managed. Through the attestation schemes and the BC infrastructure a secure device enrolment protocol must designed to deny the onboarding process to any rogue or comprised device that might pose a threat to the network. REWIRE attestation mechanisms in tandem with the blockchain infrastructure and the ZTO mechanism will guarantee the security of this process.
- **Onboarding with zero knowledge proof:** All nodes internal or external must be assessed initially during the onboarding to the network, offering verifiable evidence on the secure state of the device. After the on-boarding is completed, the agents will be still continuously evaluated to check for correct integrity, authenticity.
- **Protection from data extraction**: The binaries contained in the device's memory will be protected by the TEE and will be executed in the protected and isolated environment. In addition, the cryptographic material will be also protected by the TEE, so that to ensure that cryptographic keys cannot be extracted from the device.
- **Seamlessly integrable:** Attestation mechanisms must be seamlessly integrable into a Smart City deployment, so it doesn't interfere with the network's activity.

6.2.4. Reference scenario user stories

[ODINS.US.1]: As a Smart City System Administrator, I want to perform Firmware Updates Over-the-Air (FUOTA) over secure and authenticated communication channels so that to update the FW of end-devices to avoid technicians to perform the update physically on-site, avoiding high maintenance overhead costs. The FW must be free of vulnerabilities prior to the deployment and the end-device must be able to ensure the integrity and authenticity of the received FW.

User Story Confirmations: Updates will be validated by the SW/FW validation component of REWIRE. If correct and properly attested, the update binaries will be accepted. The status of the validation will be

reported back to the administrator. Validated updates will be eligible to perform a FUOTA. After each successful FUOTA procedure the device will report its current version to a user-friendly platform where the System Administrator can keep track of the updated devices. The update process will be based on the use of side-channel resistant authenticated encryption scheme of REWIRE in order to ensure the confidentiality and authenticity of the update.

REWIRE Functionalities: To achieve a high level of security in REWIRE, there is exacerbated needs for FUOTA features and procedures.

- Validation of firmware will be performed using the SW/FW validation component to ensure that the firmware is free of known vulnerabilities and implementation flaws.
- Use of side-channel resistant authenticated encryption scheme of REWIRE to ensure the confidentiality and integrity of the FW update process. The existing SW/FW update service of ODINS will be enhanced to work in synergy with the REWIRE artifacts.
- Isolation code/process which is in charge of performing the firmware update on the device from the main current code. Isolation will be based on the qualities of the REWIRE TEE and the use of enclaves.
- Use of REWIRE attestation schemes to ensure the integrity and the correct configuration of the deployed SW/FW update.
- Distribution of the SW/FW update through the Blockchain infrastructure and the Secure Oracles and support of different modes of operations, i.e., one-to-one (update of one device), one-to-many (distribution of an update to multiple devices simultaneously).

The following diagram illustrates the workflows which will take place among the various REWIRE components in order to realise the user story. The diagram is based on the components as those have been documented in the main REWIRE architectural diagram in chapter 4. It is expected that the diagram and the respective interactions will be updated as the project progresses.

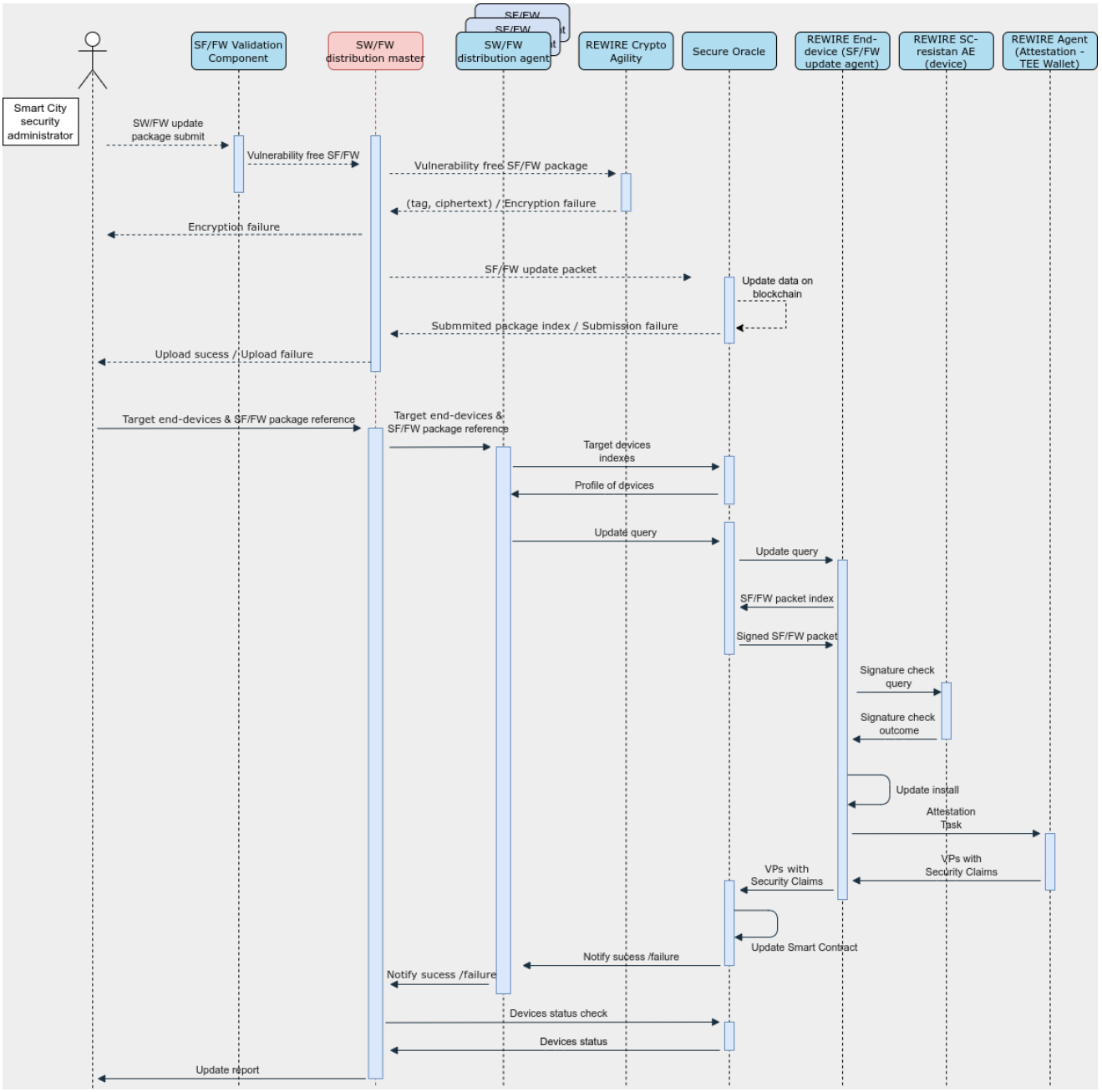

Figure 6.4: ODINS US 1 Sequence Diagram

[ODINS.US.2]: As a smart city administrator or a security administrator in a Smart City scenario, I want to easily and securely onboard new devices and avoid cumbersome deployment procedures that require per-device specific configuration. To increase security of end-device's onboarding I want to attest the integrity of the end-device without any trust assumption.

User Story Confirmations: The existing infrastructure will be enhanced with the REWIRE ZTO mechanisms to support easy and secure onboarding of devices in smart cities infrastructure, without any trust assumption and based on the principle "Never trust, always verify". The ZTO mechanism will be supported by the REWIRE blockchain architecture to be able to manage access to the REWIRE infrastructure and to manage security claims issued by the devices. Evidence on the integrity of the system will be issued as verifiable presentations and will be used to build trust-aware onboarding mechanisms. Onboarding will only be allowed if the device has proved its correct bootup and state.

REWIRE Functionalities: To achieve a high level of security in REWIRE, these are the functionalities that need to be combined to realise the underlined user story:

- Through the attestation agent, the ZTO process is being triggered based on pre-specified policies. The ZTO engages the interaction of the Privacy CA which connects with the MUD profile server and the AAA server from the manufacturer domain.
- The attestation agent is responsible of triggering the process of taking measurements of the device resources in order to generate the necessary verifiable credentials which contain the security claims for the integrity of the system.
- The blockchain infrastructure will then apply the trust-aware authentication and authorisation of the devices based on the provided verifiable presentations.
- The interactions with the blockchain infrastructure are facilitated through the secure Oracles which handle the authorisation process utilising the specific business logic, as expressed in the chaincode of smart contract.

The following diagram illustrates the workflows which will take place among the various REWIRE components in order to realise the user story. The diagram is based on the components as those have been documented in the main REWIRE architectural diagram in Chapter 4. It is expected that the diagram and the respective interactions will be updated as the project progresses.

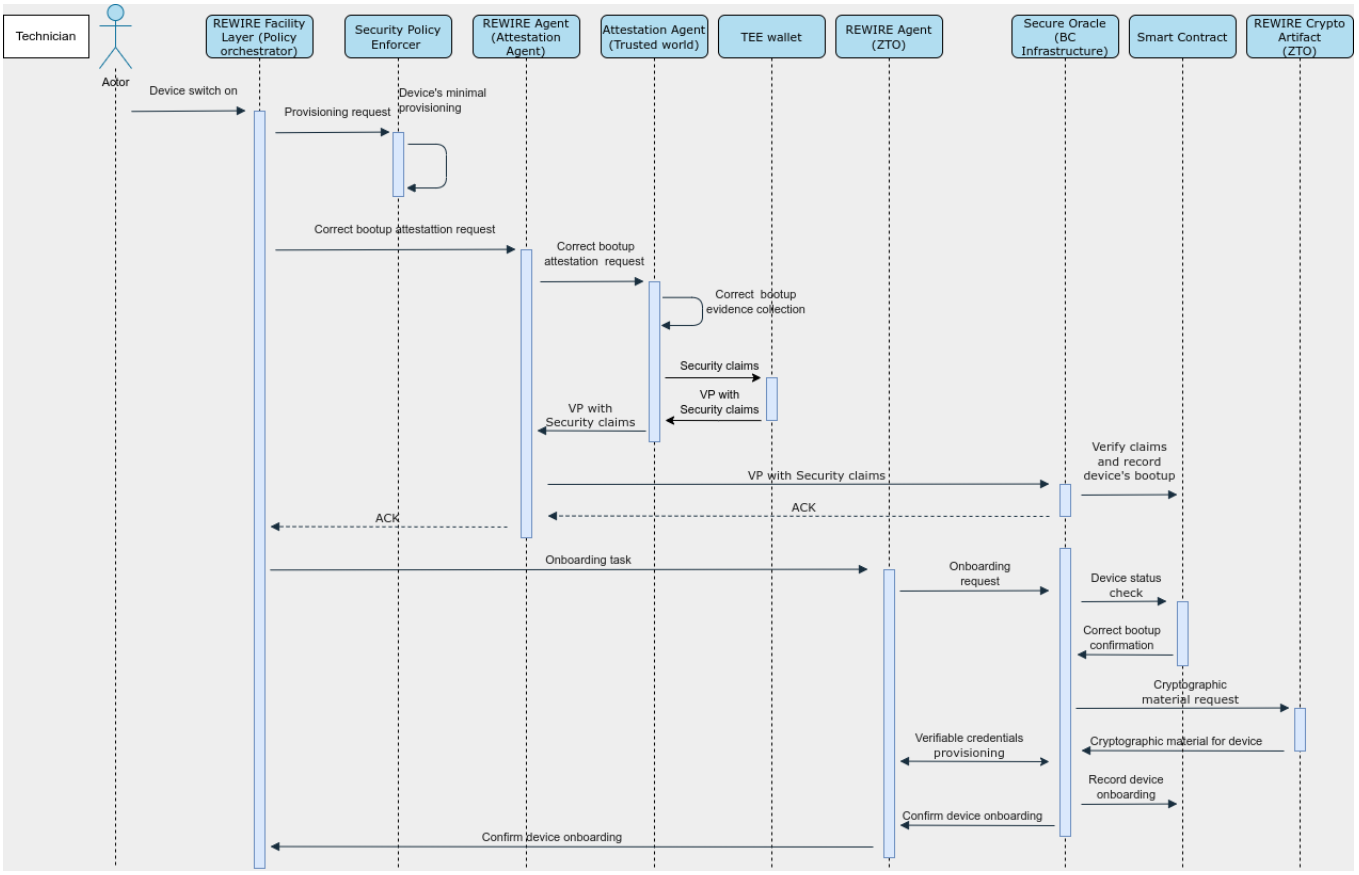

Figure 6.5: ODINS US 2 Sequence Diagram

[ODINS.US.3]: As a System security Administrator, I want to specific control flow of end-device's critical processes through previous instrumentation of the SW/FW to continuously assess the expected/regular operating behaviour of the device. The collected evidence will be used to efficiently attest device's integrity in order to raise alarms when devices are not behaving as expected or in a potentially threatening way.

User Story Confirmations: The REWIRE attestation schemes will be applied on the devices of the smart cities in order to attest in a dynamic manner either static properties of the devices (e.g., the binaries or configurations) or the execution behaviour of a critical function. The attestation mechanisms will capitalize on the REWIRE introspection artifacts, I.e., the REWIRE tracer which will be in position to introspect the execution behaviour of a function based on the hook that have been instrumented by the SW/FW validation component. If the attestation scheme detects any deviation from the expected/regular operating behaviour, and if determined to be in a potential threat, the REWIRE architecture will inform the system administrator though the risk assessment platform.

REWIRE Functionalities: In order to have an attestable verification of end-device's integrity REWIRE must provide:

- Efficient attestation schemes that can attest dynamically static and dynamic properties of the devices and be supported by key restriction usage policies.
- SW/FW pre-instalment instrumentation, using monitoring hooks, to allow for evidence collection of the execution profile of critical functions on the end-devices.
- Continuous monitoring performed over the device runtime leveraging on instrumented SF/FW and the REWIRE tracer.
- Mechanisms to visualize security events and inform system administrators about the security posture of critical functions and devices. The information will be forwarded to the risk assessment engine though the blockchain infrastructure and the use of secure oracles.

The following diagram illustrates the workflows which will take place among the various REWIRE components in order to realise the user story. The diagram is based on the components as those have been documented in the main REWIRE architectural diagram in Chapter 4. It is expected that the diagram and the respective interactions will be updated as the project progresses.

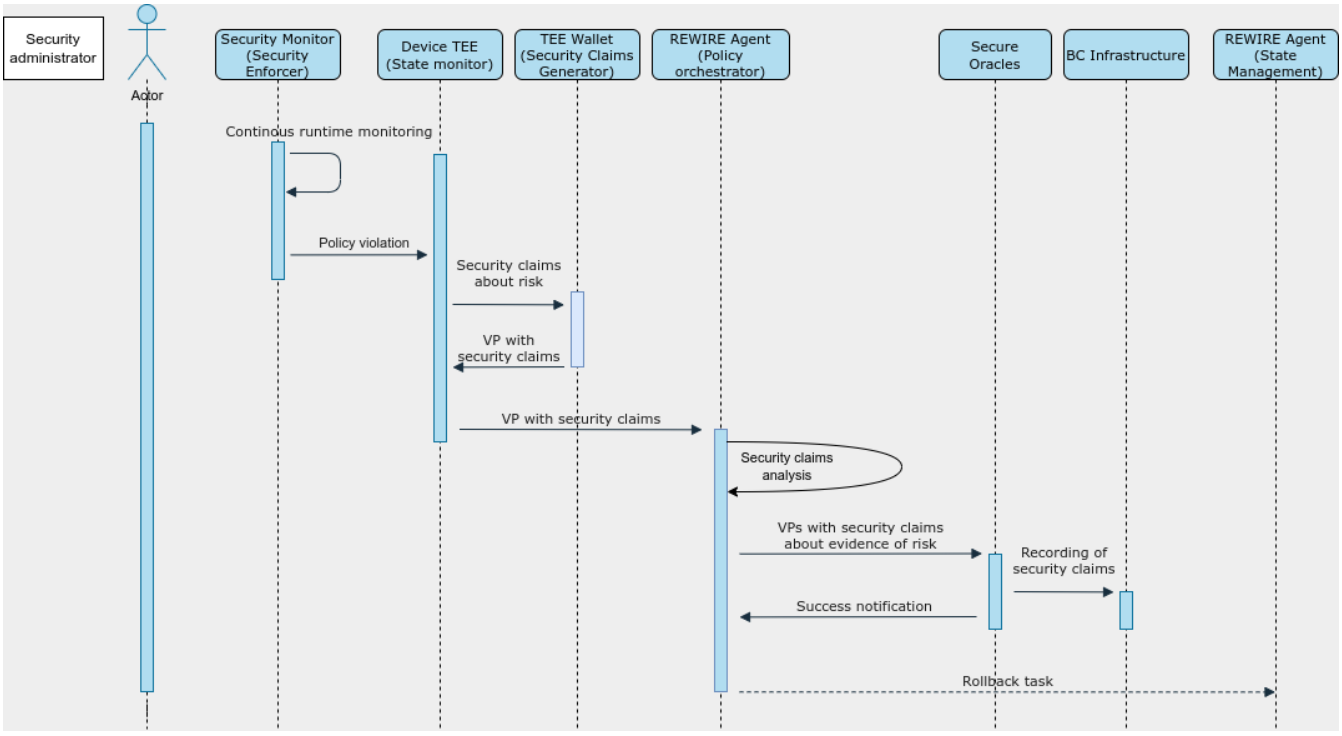

Figure 6.6: ODINS US 3 Sequence Diagram

[ODINS.US.4]: As a System Security Administrator, I want to be able to detect misbehaviours of devices in a scalable manner throughout the scattered deployments of smart cities and increase awareness on risks.

User Story Confirmations: The AI-based misbehaviour detection mechanisms of REWIRE will be utilized in order to detect misbehaviours in the highly distributed topologies of smart cities. The AI models will be fed with data stemming from the devices, through the secure oracles, which will be in position to apply data filtering and transformations so that to deliver data to the AI models in the necessary data format. The detection of misbehaviours could be an indication for threats, and the REWIRE platform will utilize the Risk Assessment framework to visualise the detected events and report on potential risks.

REWIRE Functionalities: In order to effectively detect and assess security incidents and device misbehaviour , REWIRE must provide:

- AI-based Threat Intelligence solution for analyzing system and network data so that to detect patterns of malicious or abnormal activities. The AI-based mechanism should be in position to operate over the highly distributed environment of smart cities and have scalable performance.
- Secure oracles to handle multiple data sources and apply data filtering and data manipulation before data is fed to the AI models.
- On-chain & off-chain data management utilizing off-chain data storage solution and proper data query and indexing.

The following diagram illustrates the workflows which will take place among the various REWIRE components in order to realise the user story. The diagram is based on the components as those have been documented in the main REWIRE architectural diagram in Chapter 4. It is expected that the diagram and the respective interactions will be updated as the project progresses.

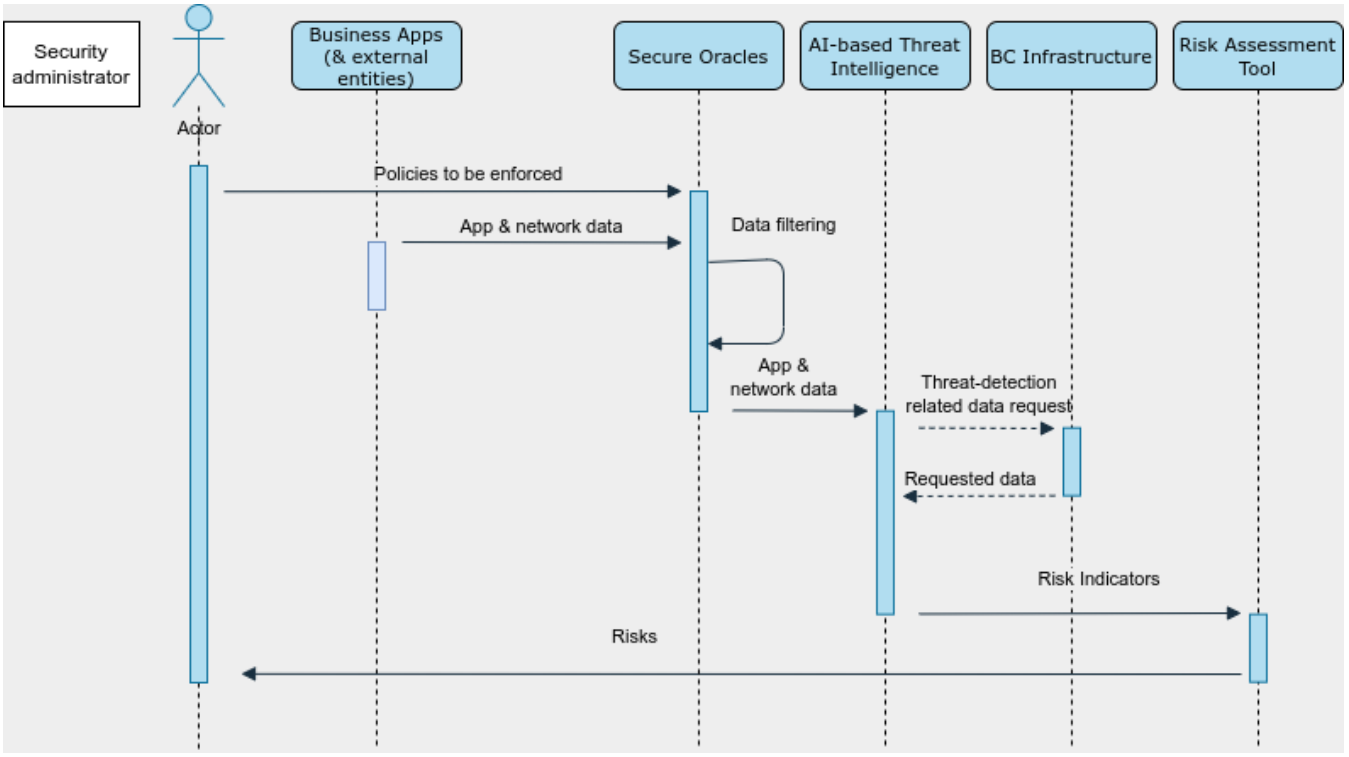

Figure 6.7: ODINS US 4 Sequence Diagram

[ODINS.US.5]: As a Security Administrator or external entity/authority I want to audit the security posture

of a smart city deployment in a secure and accountable manner. Access to collected data and security events needs to be regulated so that only authorised entities should be in position to assess the collected data, leveraging on ABAC mechanisms, control External Entities checks on critical operation's status and end-device's integrity through the centralized ledger.

User Story Confirmations: External entities must be allowed to perform checks on attested information through the BC infrastructure. They will only have access to information that is relevant to its business. Access will be controlled through ABAC managed by security oracles. These entities will be in position to verify the collected information about the security posture of the devices and will be able to validate the verifiable presentations that bear the information of the generated security claims of the devices.

REWIRE Functionalities: In order to allow External Entities to check on system integrity attestation REWIRE needs to provide:

- Attribute-based access control (ABAC) mechanism empowered by the blockchain infrastructure.
- Security oracles are able to enforce the ABAC model through the use of smart contracts.
- On- and off- chain data storage for the collection of threat intelligence data, supported by mechanisms that enable data querying and indexing.
- The generation of security claims in the form of verifiable presentations capitalizing on the use of the REWIRE attestation schemes.

The following diagram illustrates the workflows which will take place among the various REWIRE components in order to realise the user story. The diagram is based on the components as those have been documented in the main REWIRE architectural diagram in Chapter 4. It is expected that the diagram and the respective interactions will be updated as the project progresses.

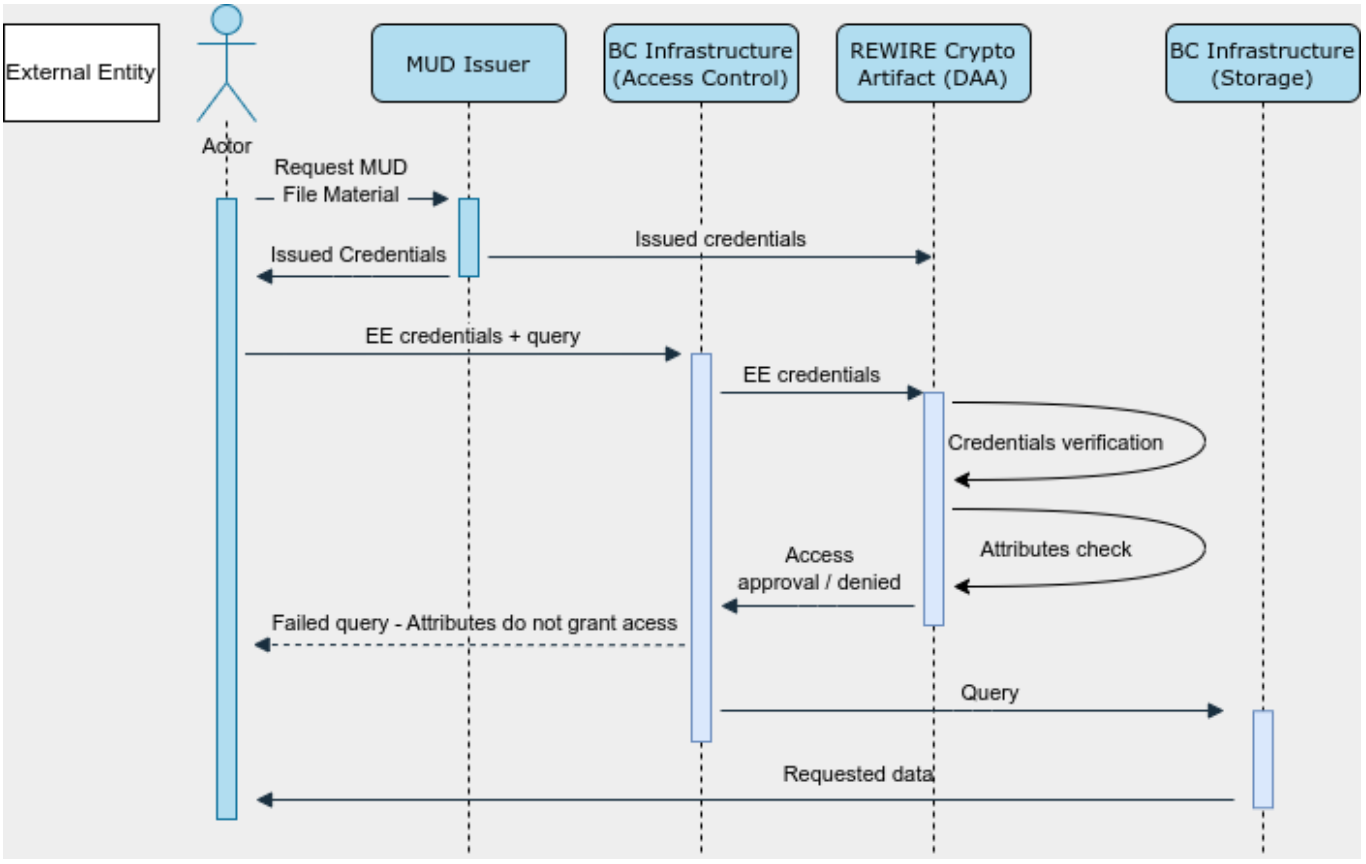

Figure 6.8: ODINS US 5 Sequence Diagram

6.2.5. Metrics of success

6.2.5.1. Quantitative Metrics

ID	Metric	Target Value Without TEE	Target Value With TEE	(M)andatory / (G)ood to Have / (O)ptional
1	**End-device's overhead of communications due to the adoption of REWIRE security controls.** (e.g., focusing on overhead of AI-based misbehaviour detection, attestation protocols, I/O for crypto key accessing.)	<15% overhead	<30% overhead	M
2	**SW/FW update critical functionality downtime**	< 20 secs	<35% overhead	M
3	**Security lifecycle management time** (Including: periodic monitoring of device state during runtime, creation of VCs, interaction with blockchain)	<1.5 min	<20% overhead	M
4	**End-device's resource usage with REWIRE mechanisms in place** (including the tracer)	<25% overhead	<35% overhead	M
5	**Risk management lifecycle time** (From the moment of the evidence collection until the moment of defining a new security policy. Including the processing time of adding monitoring hooks, but excluding the time for formal verification and the manual processes undertaken by the admin.)	4 mins	N/A	M
7	**AI-based misbehaviour detection** (including inferencing mode of the AI model and the data processing/filtering time in the secure oracles)	< 800 ms	<10% (in oracle TEE)	M
8	**SW update process (1-to-Many mode)** (Measuring the time for REWIRE security mechanisms (e.g., device authentication, access control), but excluding networking overhead factors)	< 600 ms	< 25% overhead	M
9	**Zero-touch Onboarding** (excluding network overhead)	< 700 ms	<20% overhead	M

6.2.5.2. Qualitative Metrics

ID	Metric	Target Value	(M)andatory / (G)ood to Have / (O)ptional
1	Trusted communication channels between the cloud backend and the smart cities' devices	To be supported	M
2	REWIRE Usability and integration	To be supported	M
3	Integrity protection of device configuration and behaviour	To be supported	M
4	Ease of Deployment of a trusted device	To be supported	M

6.3 Use Case – 2: Adaptive In-Vehicle SW & FW Patch Management & Software Functions Migration

Due to the rapidly increasing complexity of automotive software and the evolution of automotive technologies, such as autonomous driving (AD), advanced driving assistance systems (ADAS), augmented reality, etc., new trends arise in the field of automotive software (SW) and hardware (HW) engineering. Automotive industry-driven trends like smart mobility, connectivity, electrification and AD are paving the way for new automotive applications with extended needs in SW updates and security.

More specifically, over-the-air (OTA) updates, will be highly important for the future of vehicle connectivity. This will allow remote upgrading/fixing of vehicle functionalities, which introduces various benefits to both the automotive original equipment manufacturers (OEMs) and the customer/driver. Many recalls are due to software malfunctions and this process increases OEM's repairing expenses [REF-176]. By introducing OTA SW updates, recalls are redundant, as the SW will be delivered by the OEM straight to the fleet, group or individual cars that need an update/upgrade/bug fix. Some of the basic advantages of OTA updates read as:

- **Lower cost:** OTA updates do not require the owner to bring the car to the dealership, thus warranties are kept intact, and number of recalls is minimized.
- **Frequent updates:** OTA updates allow for a higher frequency of updates, thus keeping vehicle functionalities always up to date.

Although the advantages of OTA updates are pivoting the industry towards advancing such wireless technologies, a new opportunity for cyber attackers arises [REF-177]. SW or firmware (FW) OTA update process leaves the vehicle vulnerable to cyber-attacks, such as Man-in-the-middle attack. A potential security breach during the update process can put the driver's life at risk, therefore, the need for securing vehicle-to-OEM communication, is of high importance. Besides the attack vector concerning OTA, the attack surface is further increased due to the internal SW and HW complexity of modern cars that need to facilitate advanced functionalities. Thus, it is crucial to develop security measures and fail-safe mechanisms to face potential cyber-attacks (side-channel, spoofing etc.).

The automotive industry currently uses various common and standardised security methodologies, followed from most OEMs. Most of these have been integrated within the AUTOSAR framework [REF-178], a commonly used reference architecture. Additionally, depending on the peculiarities and preferences of each automotive system, vendor or OEM, custom security solutions are constantly emerging and can further facilitate and improve security, such as communication firewalls and intrusion detection systems. Within Figure 6.9 below, we briefly refer to some commonly used security solutions within AUTOSAR framework.

AUTOSAR (AUTomotive Open System ARchitecture) is a global development partnership of automotive manufacturers, suppliers, and tool vendors that was established to create and promote a standard for automotive software architectures. The primary goal of AUTOSAR is to standardise the software architecture of Electronic Control Units (ECUs) in vehicles, making it modular and easier to develop and integrate software from different suppliers. AUTOSAR achieves this by defining a standardised software architecture and interfaces that enable the development of reusable software components that can be used across different vehicle platforms. As a widely adopted standard in the automotive industry, it is being used to develop software, independent of hardware, for a broad range of automotive applications, including powertrain, body, chassis, and infotainment systems.

Figure 6.9 AUTOSAR's available technologies for different versions. Figure adopted by [REF-179]

There are also OEM/vendor specific security mechanisms which are not relevant to AUTOSAR but are gaining momentum in the automotive sector. Firstly, the CAN firewall is a mechanism that can be implemented in a Controller Area Network (CAN) bus system to provide protection against unauthorized access and malicious attacks. Although a promising strategy, it is not able to fully protect the whole vehicle network [REF-180]. Two typical features of CAN firewall are filtering out non-DBC defined CAN frames and CAN message period monitoring. Filtering out non-DBC defined CAN frames means that the firewall can be configured to allow only the transmission of CAN frames that are defined in the DBC (CAN database) file. The DBC file defines the structure and content of the messages that are transmitted on the CAN bus. The other capability, CAN message period monitoring, involves monitoring of the period between messages to detect any abnormal behaviour. For instance, if a message is transmitted more frequently than expected, it could be a sign of a malicious attack, e.g., a Denial of Service (DoS) attack. The firewall can detect these abnormalities and take action to prevent further attacks.

Another common security solution is the CAN Intrusion Detection System (IDS) [REF-181] which is a software or hardware system designed to monitor CAN bus traffic for suspicious activity or potential attacks. Message sniffing is required for the process of capturing and analysing network traffic to identify potential security threats. In the context of a CAN IDS, we have the following parts:

- Message sniffing which involves monitoring of the messages being sent over the CAN bus to detect any abnormal or suspicious activity.
- Message forwarding referring to the process of intercepting and redirecting network traffic from one endpoint to another. This functionality could be used to redirect suspicious messages to a separate analysis system for further investigation.
- Payload manipulation refers to the process of altering the content of a message or packet. Also, it could be used to modify the data being sent over the CAN bus to prevent a potential attack or to gather more information about the attacker.

Development of new security measures for internal and external automotive networks is an active field of research, and the need for increased security in smart and connected vehicles of the future keeps rising. For example, CAN is well known for its inability to cover security needs of more complex automotive architectures [REF-183], and that's why Flexray, and more recently Ethernet [REF-182], are taking over the market. Since the complexity of vehicles is only increasing, such an upward trend will be followed by the attack surface of vehicle systems, thus a gap, both in market and literature exists, which needs to be addressed by new technologies that ensure functional safety and seamless maintenance.

Hence, the ever-increasing complexity of the automotive stack poses the need for verified SW and HW co-designs that can guarantee the assurance-by-design quality, especially when it comes to safety critical subsystems of vehicles. Furthermore, as highlighted before, the dynamicity of the automotive ecosystem demands for fail-safe operations, making the design of OTA processes and in-vehicle protection mechanisms a necessity for guaranteeing the operational assurance and continuity of vehicles' critical systems.

6.3.1. "As-is" Scenario

Current automotive electrical/electronic (E/E) architectures follow a "decentralized" architecture, where each specific vehicular function is managed from an individual ECU, with a common bus network

connecting to various ECUs within the vehicle. This approach has been followed for many years and is currently applied on the vast majority of current production vehicles. Typical bus networks are CAN, CAN-FD, FlexRay and LIN, all in widespread use in the automotive industry.

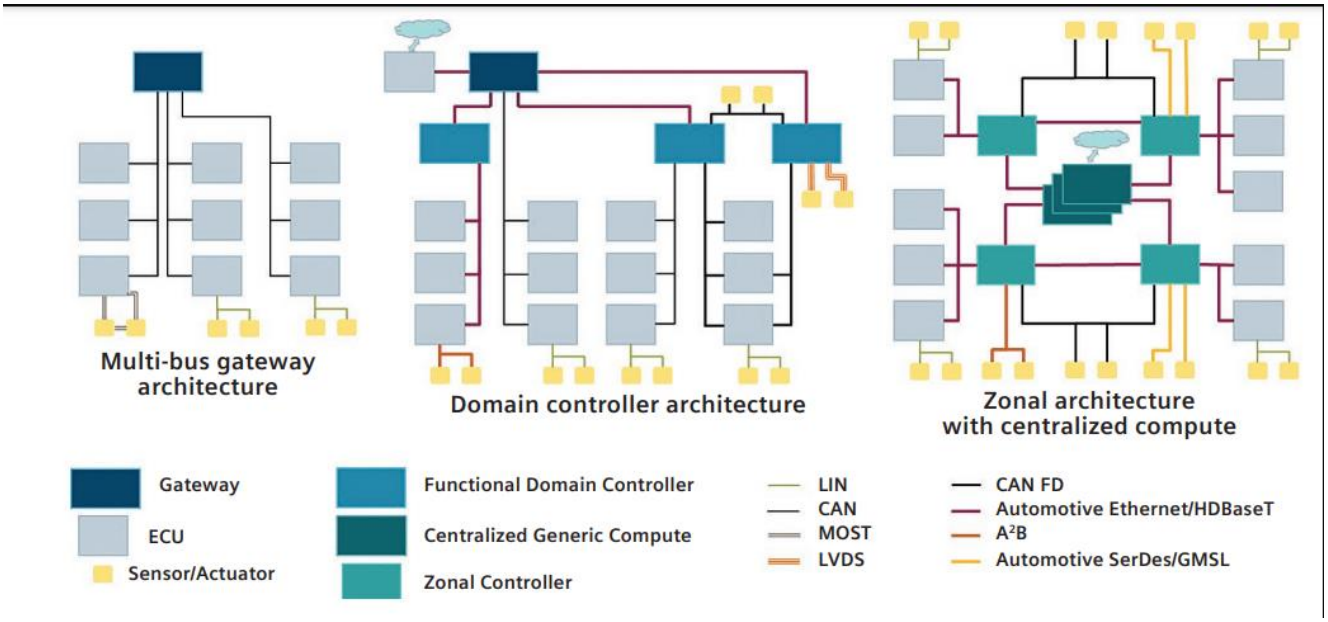

Figure 6.10: Decentralized (left), Centralized Domain (center) and Centralized Zonal (right) architectures. Figure adopted by [REF-184]

This “decentralized” approach presents many drawbacks and challenges [REF-185], primarily on scalability and communication. In terms of scalability, with each primary function grouped into a single ECU, the total number of ECUs on vehicles today can rise up to 100 ECUs creating severe communication dependencies and increased network load. Additionally, different vehicle variants (e.g., EU vs USA, entry vs premium models, conventional vs hybrid engine on the same model, etc.) require significantly different SW parameterization or functionalities as well as different I/O, HW and peripherals handling. This creates a necessity for SW/HW variation on a wide range of ECUs, typically implemented from different Tier1 suppliers or vendors for the same vehicle, creating significant workload to OEMs.

In light of these challenges, “centralized” ECU network architectures are gaining strong momentum with significant number of OEMs/Tier 1 suppliers researching these solutions. In contrast to decentralized approach, centralized architectures group more vehicle functions into a single ECU, in order to decrease network load and minimize total number of ECUs. As an example, on a centralized approach, powertrain functions would be grouped into a single ECU, while a decentralized architecture would have individual ECUs for each function (Transmission Control Unit, Engine Control Unit, Battery Management System, etc.). A fully centralized approach would have all vehicle functions grouped into a single, “master ECU”, with various “Zonal Controller Units” (ZCUs), handling only a very primitive logic to acquire and re-transmit sensor/actuation data.

Furthermore, with current automotive trends moving towards connectivity and autonomous driving, the increased security needs (e.g., data encryption or authentication of bus messages) create even more communication load, which on many cases is unfeasible to implement on the current ECU processors, decentralized architectures and network protocols like CAN bus.

For this reason, the automotive industry moves more and more towards higher bandwidth communication protocols such as CANFD and FlexRay. While indeed CANFD and FlexRay offer increased bandwidth, the problem to some extend remains, while FlexRay also presents significant drawbacks, i.e., each communication dependency needs to be known in advance during development. Latest automotive trends are moving progressively to Automotive Ethernet, to significantly increase bandwidth and provide a better communication medium in which to implement a secure ECU communication [REF-185].

Traditional ECUs typically handle and control a dedicated and conventional mechanical sub-part of the vehicle system (engine, brakes, suspension, etc.) that usually does not change or require any significant update within the vehicle lifecycle. Thus, SW within those ECUs is typically only on occasion updated, usually at a local dealership at each periodical vehicle maintenance primarily through the OBD port, with AUTOSAR defining Secure Logging and Diagnostics via UDS services 27 and 29 that can achieve user-2-device (i.e., dealership tool-2-vehicle ECU) authentication.

In contrast, new automotive functionalities that are expanding towards V2X connectivity, in-vehicle augmented reality, user experience, infotainment systems and autonomous driving, by definition would require significantly more SW update capabilities. This means that ECU SW is becoming an alive and active product throughout the vehicle lifecycle, forcing the need for constant SW updates, either for new SW functionalities, parameterization or for bug fixing and vulnerability patching.

As an example, Neural Networks, an integral part of Autonomous Driving systems, are required to be constantly trained, updated and tailored for specific edge cases and road environment (urban vs rural roads, high density and narrow city streets vs highways, and so on) since the OEMs cannot perform all case analysis in advance. Additionally in the context of advanced driver-assistance system (ADAS)/ Autonomous Driving (AD), OTA and central server communication is needed to receive new HD (High-Definition) Maps as the vehicle is driving, something very important for higher levels of driving automation. Similarly, in the near future, new maps will be created constantly, depicting a construction zone, new traffic signs that have been just installed and many more.

V2X communication is as well of vital importance since vehicle communication with road entities (traffic lights, other cars, etc.) will allow transmission of critical information for the safe vehicle operation and traffic monitoring. Typical V2X scenarios currently examined from the automotive industry include communication between a given vehicle and an adversary vehicle as well as the on-boarding of a vehicle (ex. when it is powered-on) to a given OEM fleet network or to a government-regulated network. Such communication and vehicle on-boarding presents security challenges since it requires validating or authenticating each vehicle upon the establishment of the communication.

Therefore, OTA updates and in general over-the-air communication with road/infrastructure entities is a necessity and will be vital for successful and secure ADAS/AD systems operation. Currently, these systems are more “hard-wired” within the vehicle or non-present at all, limiting vehicle capabilities. While indeed most cars today do not have significant or extended OTA capabilities or have none at all (most OTA cases can be found on Infotainment systems, thus noncritical functions/devices), according to Market Research Future [REF-186] , the automotive OTA updates market is expected to grow 18% from 2022 to 2030. As a result of OTA necessities, massive security concerns are emerging, since potential vulnerabilities or security weaknesses can significantly impact road safety.

From common standardization perspective, AUTOSAR defines usage of IPsec/TLS for automotive OTA updates. Below follows an analysis [REF-187] of an exemplary OTA setup.

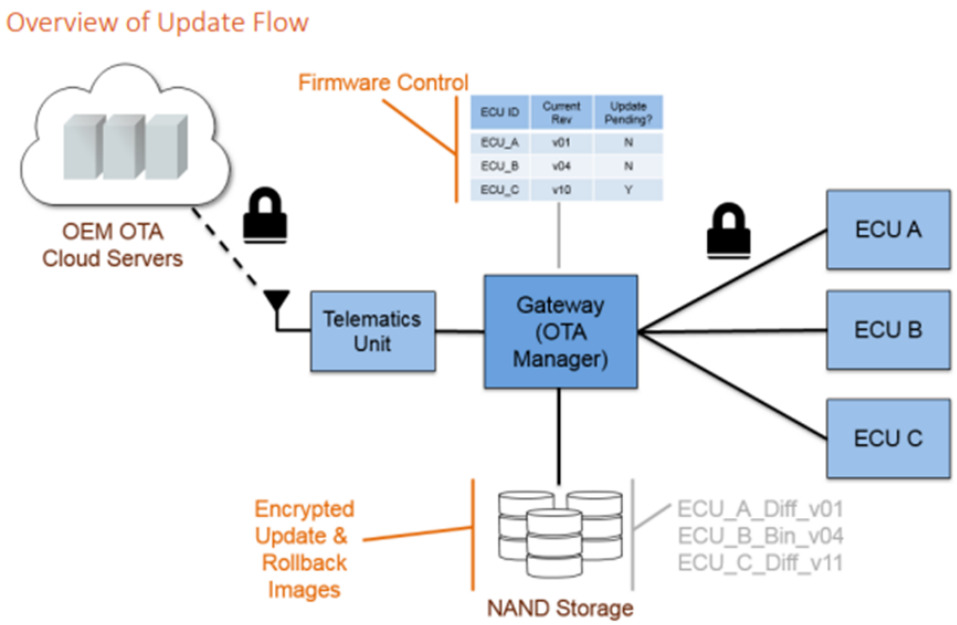

Figure 6.11: Key components of an OTA system. Figure adopted by [REF-187]

Generally, the Telematics Unit is responsible for establishing the connection to the OEM server. This unit can employ TLS, to secure the communication over the mobile network. Once the secured update file is received, it is processed by the Gateway (OTA Manager) through a table which contains information about each ECU in the vehicle, including their serial numbers and current firmware versions. This enables the OTA Manager to verify the compatibility of the update with the vehicle.

A basic assumption is if the targeted ECU supports firmware decryption and authentication. In such a case, the update file can be sent directly from the OTA Manager. However, **not all ECUs have secure key storage and hardware accelerated security**. On such occasions, the OTA Manager needs to authenticate and decrypt the update on behalf of the target ECU and then transmit the update over the internal network. As a result, OTA Manager plays a crucial role in verifying the validity of an update for the vehicle to ensure the consistency and compatibility of firmware across all ECUs in the vehicle and to prevent an attacker from installing an older firmware version to exploit known vulnerabilities. However, in such cases the communication between the OTA and the corresponding ECU remains unprotected as the secure communication, OTA validation and signature verification take place on the OTA Manager and not on the actual ECU.

Within a centralized architectural approach, as more and more functions are grouped into a single ECU, together with the increased security needs of V2X and autonomous driving technologies, fail-safe mechanisms and roll-back or mitigation measures emerge with increased necessity, as each potential ECU compromise (primarily due to a security compromise but the concept holds true for safety as well) can impact a significantly increased vehicle functionality. Hence, to ensure the continuity of critical functions there is a demand for “vehicle functions migration” to alternative, auxiliary or backup ECUs, in order to increase the overall system reliability, robustness and security.

While the details, peculiarities and complexities of various proposed “centralized” architectures are indeed very extended and a topic of significant automotive research, these fall beyond the scope of REWIRE project. Therefore, the automotive use case will focus into exploring this concept on an exemplary centralized architecture.

Given the analysis of the current state of the automotive domain and KENOTOM’s vision for integrating secure solutions which will enhance the security and operational assurance of safety critical automotive operations, the next section defines the challenges that REWIRE project needs to consider.

6.3.2. Scenario's Challenges and Needs from REWIRE

Given the description provided in the previous section, it becomes clear that despite the benefits of the "centralised" architecture, several challenges emerge.

- **Increased attack surface**: In general, autonomous vehicles have more exposed entry points for malicious cyber-attacks, due to their increased complexity [REF-188] along with the different type of sensors. Connected Autonomous Vehicles (CAVs) will be even more vulnerable from a security perspective, as more electronic equipment is needed for implementing systems like ADAS, LKAS etc. So, there is necessity for additional security mechanisms to achieve operational assurance on vehicles.
- **Validation of SW/FW updates:** Since the automotive use-case is investigating OTA SW/FW updates, it is crucial that the update package will remain secure. The SW and FW updates are not initially validated. This means that updates that may be massively applied to the vehicle may render the whole infrastructure vulnerable and can potentially impact the safety. Based on the Upstream's research reports [REF-189] about the automotive cyber incidents occurred in 2020 & 2021, 87.7% threats are related to vehicle data/code, 50.8% potential vulnerabilities that could be exploited if not sufficiently protected or hardened, 24.1% threats regarding back-end servers related to vehicles in the field, 4.3% threats to vehicles regarding their update procedures. Lastly, the update process should be followed by scrutinizing each update.
- **Regulatory and standardisation compliance for OTA and confidentiality of communications:** Autonomous vehicles are subject to regulatory frameworks and safety standards. For instance, OTA updates need to comply with industry standards, which may include specific validation and certification processes. As it is mentioned in SAE J3061:2016, "Cybersecurity Guidebook for Cyber-Physical Vehicle Systems", encryption should be used to ensure the confidentiality of the OTA update package. These standards could be utilized to establish best practices and guidelines for ensuring the robustness and resilience of autonomous vehicle systems by covering areas such as functional safety, risk assessment, secure communication protocols, software development processes, and vulnerability management. Also, regulations may differentiate per country, but they cover areas such as emissions regulations, cybersecurity requirements, data privacy, and more. That is, apart from the SW/FW update per se, we need to find a way to ensure that a transmitted SW update is confidential, and it has not been tampered by any external entity that may interfere with the channel.
- **Limitation of bandwidth:** Limited bandwidth or slow data transfer rates (i.e., CAN bandwidth 1Mbps) can restrict the security options. Ensuring efficient data transfer is crucial to minimise update time and optimise the user experience.

The aforementioned challenges, combined with the current status of the automotive landscape, lead the following needs:

- **Need for SW/FW update for safety critical functions**: Automotive systems rely on SW/FW to control various functions, including engine management, braking, steering, and driver assistance systems. Updates can address security vulnerabilities, fix bugs, and improve the performance and functionality of the system. However, updating safety-critical functions poses unique challenges due to the potential risks involved. One of main challenges is to ensure that the update does not introduce new vulnerabilities or affect the safety and reliability of the system. This requires extensive testing and verification to ensure that the update is compatible with the system's hardware and software components, and that it meets the required safety standards.
- **Need for adoption of automotive ethernet:** The limited bandwidth of the CAN bus is a bottleneck which may lead to latency issues and restrict the ability of the system to handle higher data rates required for ADAS, autonomous driving and in-vehicle infotainment systems. To address this challenge, the adoption of Ethernet-based communication protocols is mandatory. Automotive Ethernet offers higher bandwidth, lower latency, and greater flexibility than the CAN bus. It can support data rates of up to 10 Gbps and enables the integration of various communication protocols, such as audio, video, and data transfer. In addition to higher bandwidth, automotive

Ethernet also offers improved security features, including encryption and authentication, to ensure the confidentiality and integrity of data transmitted over the network. This is essential for safety-critical systems, where any interference or malicious attacks on the communication network can have severe consequences.

- **Need of function migration:** When an ECU is compromised or is misbehaving, there is the need to ensure the continuity of the critical services. The compromised functionality should be migrated to a neighbouring back-up ECU. In this way, we can guarantee the reliability and the continuity of safety-critical services.
- **Need for advanced and formally verified crypto:** To guarantee the confidentiality and the authenticity of SW/FW updates, there is a need for strong and formally verified cryptographic schemes. The update package needs to be signed using a cryptographic key, and the vehicle verifies the signature before installing the update to ensure that it has not been tampered with during transit.
- **Need for verifiable evidence advocating the secure state of vehicles:** In direction of having continuous and trust-aware authorization of vehicles, the security status of critical functions of the vehicles needs to be validated, especially when those belong to collaborative fleets where information is exchanged in the context of critical services like the autonomous driving case. Such a solution can facilitate the V2X communication needs for validation of the security state of each vehicle as it is on-boarded to either a common fleet network or to bi-directional communication with another vehicle. V2X communication needs to be supported by advanced solutions which can validate the security state of each vehicle as it is on-boarded to either a common fleet network or to bi-directional communication with another vehicle.

6.3.3. “To-be” Scenario

As mentioned above, current as-is of automotive E/E architectures pivots industrial trends towards centralization. In this section, the automotive use-case specific exemplary setup, which also includes REWIRE artefacts, will be discussed. Along with that, some possible configuration options will be presented, all of whom will serve the purpose of showcasing REWIRE capabilities through a demo. Furthermore, REWIRE framework security proofs and ECU state monitoring will be discussed.

On this exemplary setup, Zonal Controllers are serving the role of collecting and performing an initial primitive processing on data from a given sub-network of low-level ECUs. These ECUs can act as lower level, simplified “smart actuators” for example VCM (Vehicle Control Module) handles to operate the vehicle engine, or to collect brake pedal position, with the ADAS ECU used to collect perception data. Following that setup, the total computational and functional vehicle tasks could be performed on a higher level, “master ECU”, that would serve as a central vehicle computer, or even as a server to off-load the computational tasks on the cloud.

A potential security attack on a given Zonal Control Unit 1 could compromise a significant portion of the underline sub-system posing increased problems in comparison to a decentralized architecture. To compensate that, REWIRE proposes that ZCU1 functionalities could be migrated to a neighbouring ZCU2, or to the “centralized”, High-Performance Computer (HPC) as well, to either maintain some vehicle functionality or to handle the deployment of a fail-safe mechanism. Current vehicles do not have such a concept in any kind of form, since typically, if a specific ECU is compromised (e.g., system malfunction), safety SW intervenes to drive the system to a “minimum”. ECU safety SW intervention though could be also compromised at a successful or extended security breach.

In order to perform a successful migration of functions, access to the underline network would be required, which is by itself a complex topic to generalize. Since most vehicle ECUs or potential ZCUs serve a specific purpose and the cabling itself is quite extended on vehicles, not all ECUs have access to all other ECUs or specific components, sensors and actuators. Therefore, the migration of the functionality of a given ZCU 1 presumes that the alternative ZCU 2 (which is supposed to undertake the computational and functional load of the compromised ZCU 1) actually does have access to the needed network infrastructure of ZCU 1. It is in any case beyond the scope of the project to concretely conclude a realistic setup across multiple vehicle functions or network architectures and therefore this solution can only be

examined on a proof-of-concept basis within REWIRE. Nonetheless, accessing the underlying network could be facilitated through a variety of potential solutions. A first solution would be to use an auxiliary cable harness connecting ZCU 1 and 2, providing access to an isolated I/O interface with the corresponding sub-network (Figure 6.12: Option 2). Alternatively, if ZCUs hold significant portion of calculation load of vehicle functions (acting as “Domain controllers” for all sub-functions of a vehicle system such as Powertrain-Motion-Propulsion domain, Chassis-Advanced Driving-Comfort domain, etc., a variation of centralized architectures) it could be argued that each one of them could have an isolated resource (e.g., extra cores or processors) dedicated to undertake the additional work-load of the migrated functions (Figure 6.12: Option 1).

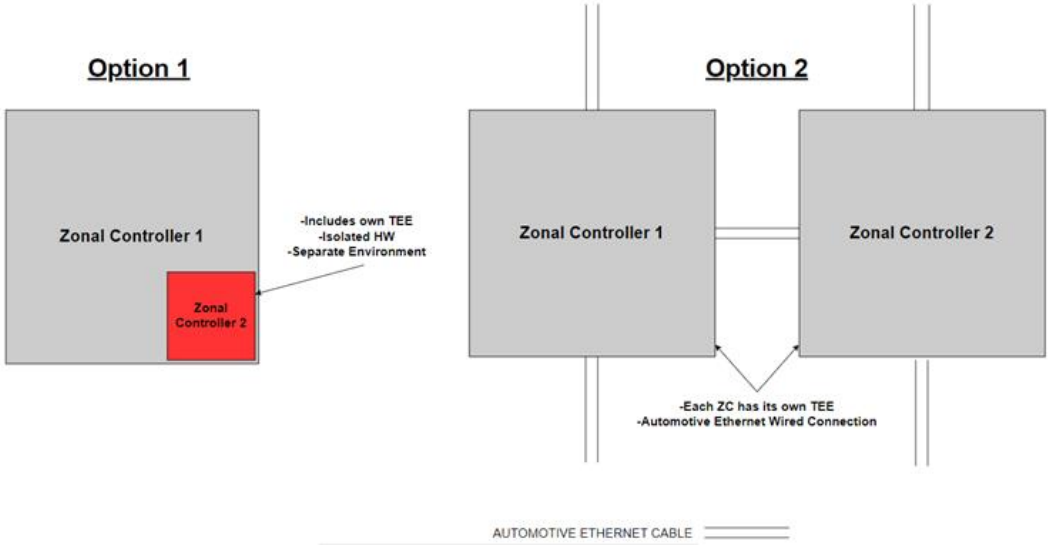

Figure 6.12: Options 1 and 2 on connectivity between Zonal Controllers

In Figure 6.13, a sketch of the demo setup along with its components is presented. Zonal/Domain Controller 1 (ZCU1) is the main on-board REWIRE edge device that will communicate with REWIRE artefacts. ZCU 2 is the second REWIRE-specific board (most likely GENESYS II board will be used) that will demonstrate migration of functions. Migration will happen by direct connection of ZCU1 and ZCU2, through the REWIRE TEE of each ZCU, by establishing a security key between the TEEs. Security of the migration will be independent of how secure the physical network is, since REWIRE TEE will undertake this task. This demo setup (Figure 6.13) could be thought of as part of a hypothetical vehicle-internal architecture as depicted in Figure y. In this architecture, ZCU2 would be primarily responsible for a different domain (ECU2, ECU3, ECU4) than that of ZCU1 and could potentially hold the migration load. In Figure y, some exemplary signals of a theoretical ADAS application are also depicted.

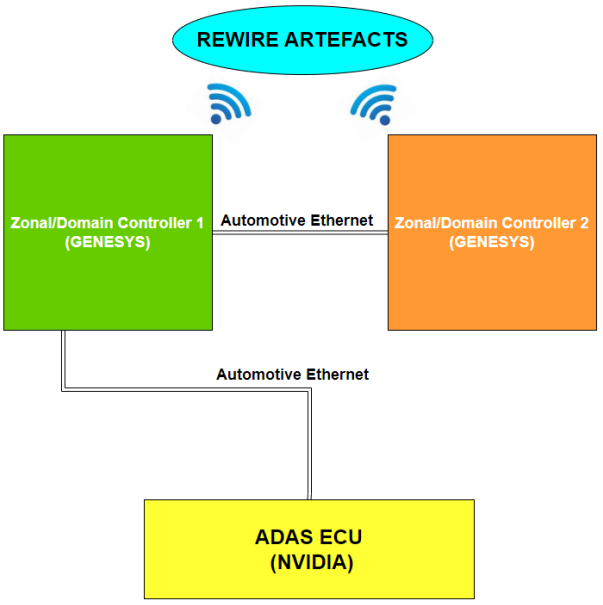

Figure 6.13: Automotive use case demo setup

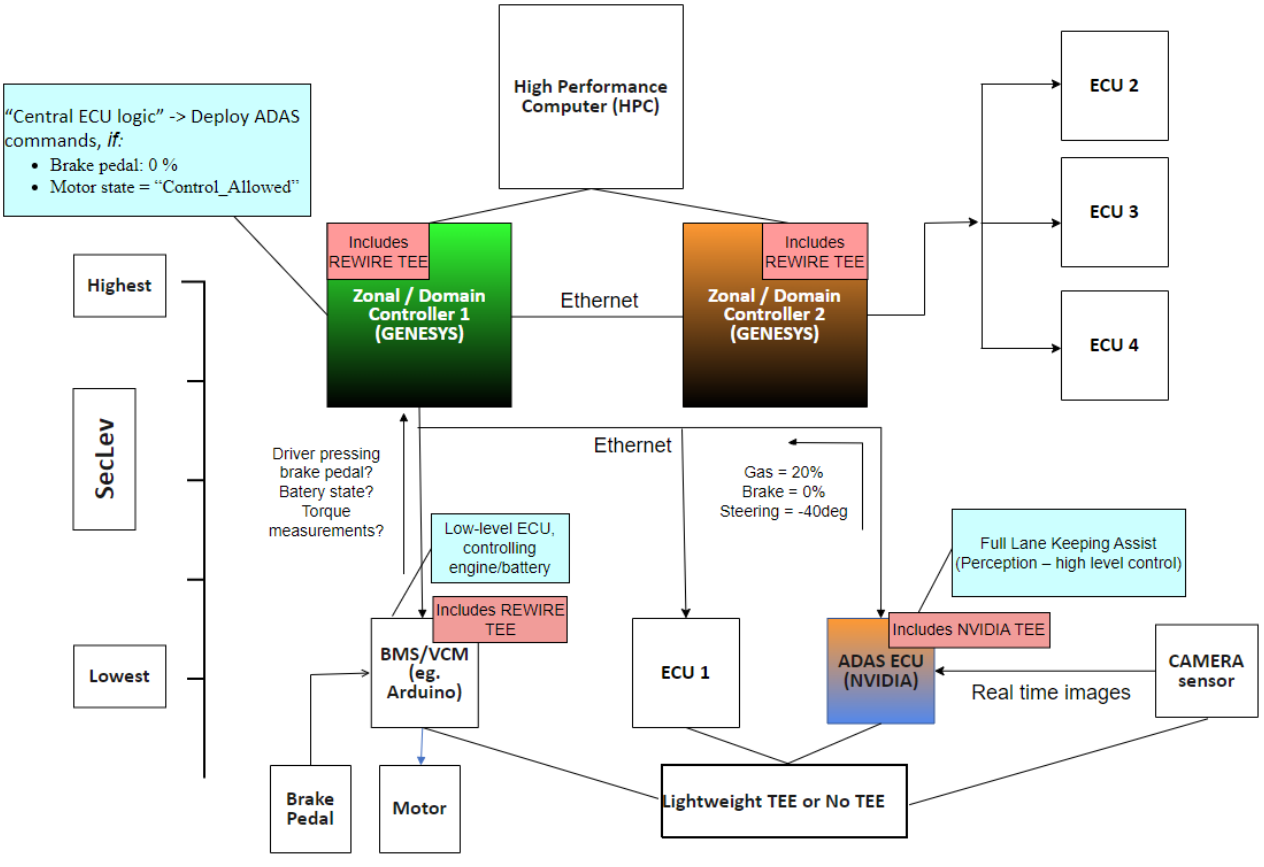

Figure 6.14: Exemplary vehicle network including automotive use case demo (example 1).

If still, no option to migrate the compromised vehicle function itself would be possible, the concept even so has validity, to act as a deployment of a fail-safe vehicle reaction. In this case, although no functionality migration is possible, transferring and deploying the compromised SW on a neighbouring controller would allow the deployment of the fail-safe mechanisms of the attacked system itself (driver warnings, emergency braking, V2X communication, etc.), but on a "clean" and secure environment. Such an exemplary case is depicted below.

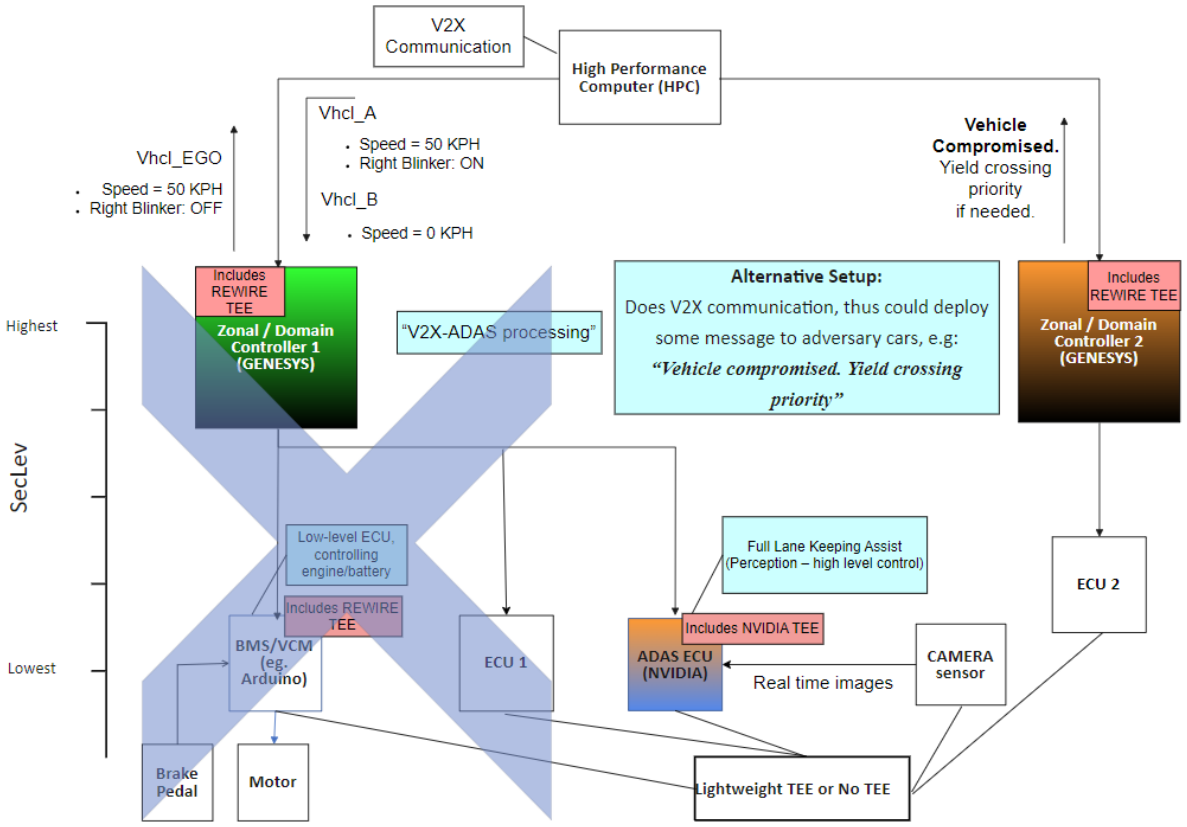

Figure 6.15: Exemplary vehicle network including automotive use case demo (example 2).

On the setup illustrated in Figure z, an exemplary V2X communication is depicted, with the High-Performance Computer (HPC) receiving V2X data (adversary vehicle speeds, positions, etc.) and transmitting them to ZCU1 (Zonal Controller 1) for subsequent usage at the ADAS ECU. Similarly, ZCU1 handles transmission of speed, motion, etc data of the vehicle itself to the HPC for informing adversary vehicles. Here, we depict a potential successful compromise on ZCU1 or/and a portion of the underlined ECUs (crossed-section area). Such a case could happen for instance through a successful manipulation of network messages on VCM or ZCU1 to constantly accelerate the vehicle. Upon detecting such an abnormality, a function migration to the neighbouring ZCU2 is deployed and due to an inability of gaining access to the engine itself, an appropriate message is transmitted to the adversary vehicles, to notify for the malfunction ("Vehicle compromised. Yield crossing priority, if needed"). The "X" crossed section over the ZCU1, in the figure, implies that ZCU1 is detected as compromised, and the function migration will be triggered.

For the Automotive Use Case and in order to minimise overall complexity of the needed HW, SW and implementation effort, REWIRE will utilize and deploy an exemplary setup, adopting a REWIRE-specific board (most likely GENESYS II board), as well as NVIDIA DRIVE ORIN as the ADAS ECU which will be responsible for an ADAS related function (Lane Keeping System or Adaptive Cruise Control). More details on the setup of the demonstrator will be reported in REWIRE D6.1 Integrated framework (1st release) and use case analysis.

Exemplary investigation on possible security attacks specifically for the Automotive use case demo (Lane Keeping System or Adaptive Cruise Control) could be optionally examined. Such automotive relevant specific attacks are threats to the internal vehicle communication integrity or attempts for manipulation of ADAS vehicle functions that can alert the signature of critical binaries or configuration files. Therefore, the automotive demo application could potentially examine ways of detecting such attacks. Depending on the overall complexity of the implementation, this exemplary investigation will be optionally explored either by using the already described REWIRE attestation schemes, or by extending REWIRE framework by

applying simple logic defence mechanisms for such ADAS or automotive detection of security breaches. Since in any case the specific automotive peculiarities fall beyond the scope of REWIRE project, if implementation of such detection logic is significantly complex, then on an exemplary case such scenarios could be assumed to be detected and taken as-is for the sake of demonstrating the REWIRE proposed counter measures (i.e., SW updates and vehicle migration) on top of the already defined REWIRE security use case. Specific attack cases will be explored and documented in the context of the WP6 deliverables.

The above-mentioned testbed setup (see Figure 6.13), will be used in order to enable the evaluation of core REWIRE solutions and help KENOTOM to meet its future objectives via: (i) Validation of a system's components, (ii) SW/FW update to patch vulnerable components, (iii) Attestation policies with key restriction usage, (iv) Attack mitigation through TEE state migration.

6.3.4. Reference scenario user stories

The validation of the to-be scenario that was detailed in the previous section will be based on the following User Stories. The entirety of the testbed and the REWIRE envisioned functionalities will be put in the context of the following stakeholder user stories that will guarantee that the experimentation and evaluation of the REWIRE mechanisms will take place with the boundaries of realistic cases that cover the needs of end users, while in parallel cover the whole spectrum of the functionalities to be validated, following the objectives of the project.

[KEN.US.1]: As an OEM system administrator, I want to ensure security on over the air (OTA) software updates, without the need for vehicles' on-site service.

User Story Confirmations: The SW update package will be checked in terms of integrity, confidentiality and authenticity using the secure mode of operation of the AES designed in the context of REWIRE. The SW update will be admitted only if it carries the appropriate security credentials, defined by the REWIRE security framework. Failure in any authenticating process, will be reported back to the administrator, so a cyber-attack or a malfunction can be detected quickly and eventually prevented. This user story will demonstrate a secure OTA update on a device in a 1-to-1 manner, i.e., the SW provider will perform the update directly to one vehicle. Prior to the deployment of the SW update the REWIRE frameworks must guarantee the update is free of vulnerabilities capitalizing on the SW/FW validation processes.

REWIRE Functionalities: In order to achieve a high level of security in REWIRE, there are exacerbated needs for OTA features and procedures.

- Differentiate the code in charge of performing the firmware update from the main current code. Thus, code isolation is needed using REWIRE TEE.
- Encryption and authentication are needed for securing the OTA update process. The side-channel resistant mode of operation for the authenticated encryption AES scheme of REWIRE will be used.
- Attestation mechanisms are needed to enhance the security and operational assurance by validating the state of the updated component.
- If attestation fails, the device will transmit the status of the failed attestation to the risk assessment component of REWIRE to increase administrator awareness on possible attacks.
- Only sanitized update packages will be deployed, i.e., the update will pass through the SW/FW validation processes to be checked for common vulnerabilities and implementation flows.
- The update process will be "one-to-one" with respect to the communication of the administrator with each vehicle, since every vehicle has its own keys and attestation properties.

The following diagram illustrates the workflows which will take place among the various REWIRE components in order to realise the user story. The diagram is based on the components as those have been documented in the main REWIRE architectural diagram in Chapter 4. It is expected that the diagram and the respective interactions will be updated as the project progresses.

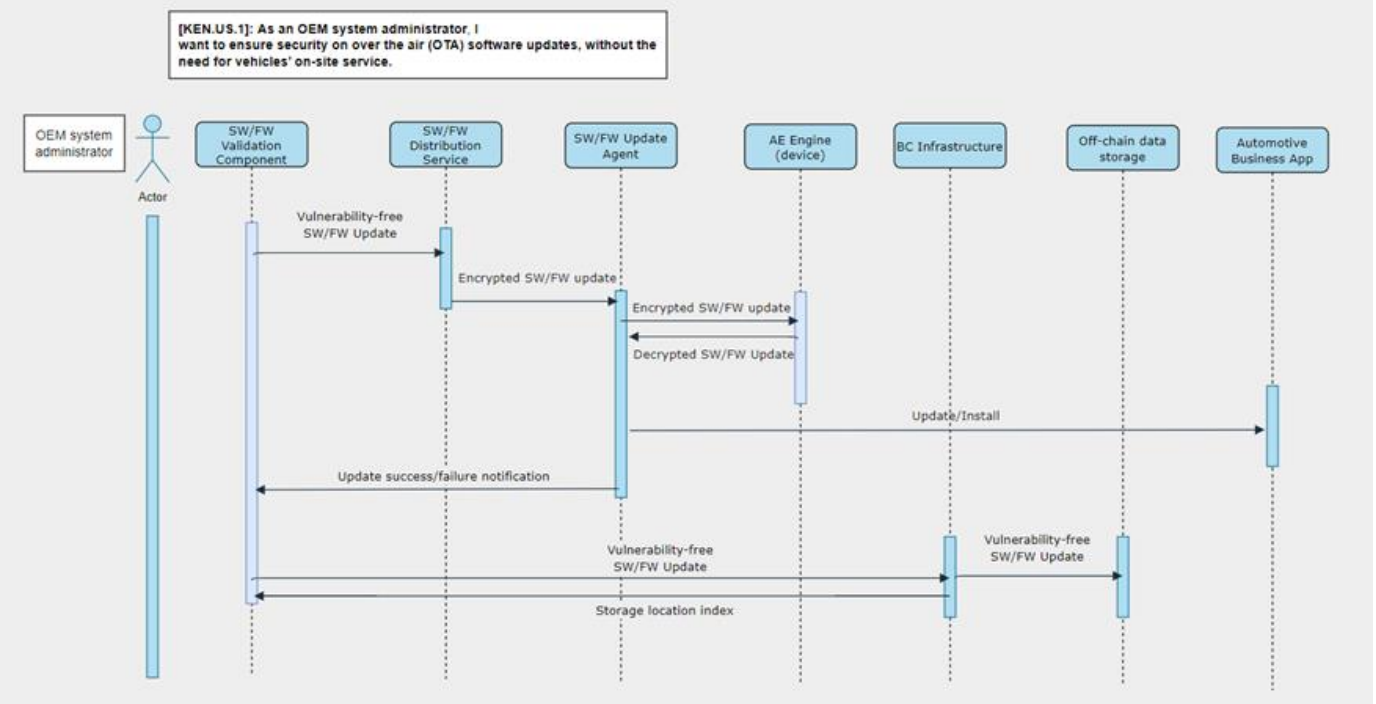

Figure 6.16: KEN US 1 Sequence Diagram

[KEN.US.2]: As an OEM System Administrator, I want to ensure that ECUs are continuously checked in order to assess their operating behaviour, and trigger alerts upon abnormal behaviour or security incidents. When an ECU detects suspicious activity, relevant stakeholders should be notified promptly to initiate appropriate response measures.

User Story Confirmations: When an ECU is compromised, the security thread will be analysed from the OEM (technicians/engineers) in order to deploy a SW fix or patch as counter measure on the attacked car or on other cars of the same brand/model to prevent subsequent attacks. To achieve this goal, REWIRE attestation mechanisms will be used in order to check the integrity of the critical system and identify potential compromise of the system. The REWIRE framework should be in position to increase the awareness of the responsible engineers.

REWIRE Functionalities: Failed attestation of the ECU will trigger an alarm that would inform the responsible engineers of the potential compromise or malfunction, to deploy SW/FW OTA updates to the corresponding system. The failed attestation event will be forwarded to the risk assessment component of REWIRE through the secure oracles.

The following diagram illustrates the workflows which will take place among the various REWIRE components in order to realise the user story. The diagram is based on the components as those have been documented in the main REWIRE architectural diagram in Chapter 4. It is expected that the diagram and the respective interactions will be updated as the project progresses.

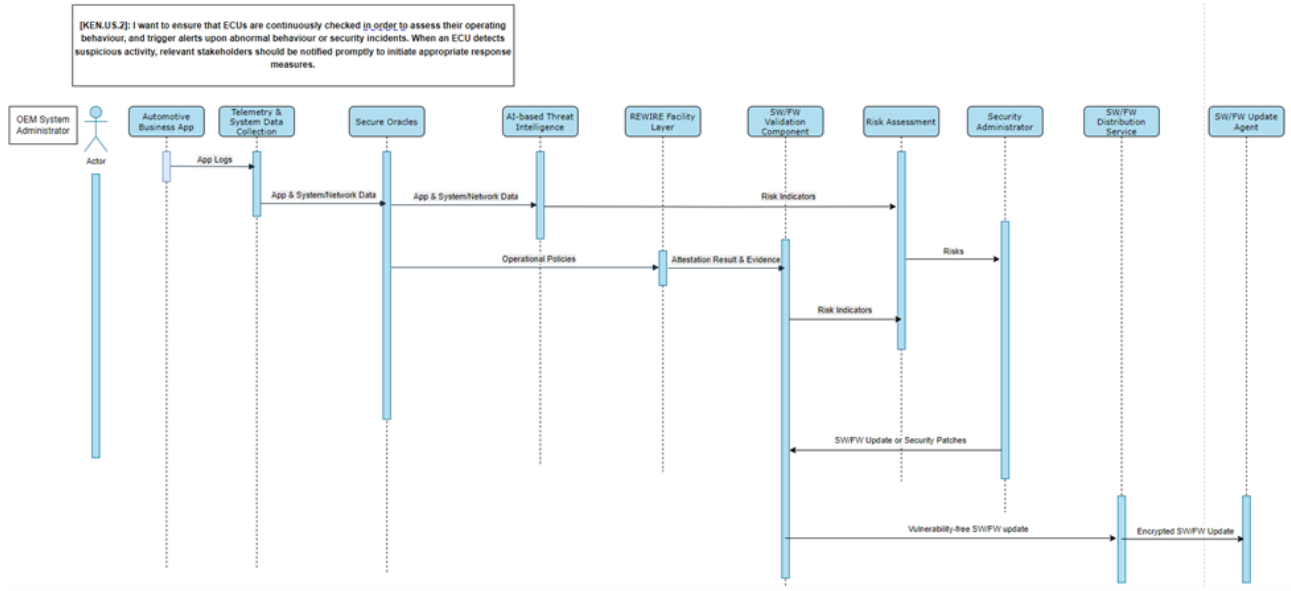

Figure 6.17: KEN US 2 Sequence Diagram

[KEN.US.3]: As an automotive engineer, I want to ensure that in the case of a security breach on a given ECU, partial or full vehicle functionality is maintained.

User Story Confirmations: When an ECU is compromised, a neighbouring ECU will be dynamically selected for function migration in order to mitigate the attack. A scenario that includes a compromised ECU will be demonstrated to test the REWIRE functionalities. Capitalising on the capabilities of REWIRE customisable TEE (based on keystone), mitigation actions will take place to enable the migration of the state of a critical function to another TEE in the vehicle.

REWIRE Functionalities: REWIRE selects an appropriate neighbouring ECU with available computational resources. Safety critical functions are migrated to the neighbouring (non-compromised) ECU. The migration operation is described as part of a policy in the REWIRE stack, capitalising on the policy enforcement mechanisms of REWIRE.

The following diagram illustrates the workflows which will take place among the various REWIRE components in order to realise the user story. The diagram is based on the components as those have been documented in the main REWIRE architectural diagram in Chapter 4. It is expected that the diagram and the respective interactions will be updated as the project progresses.

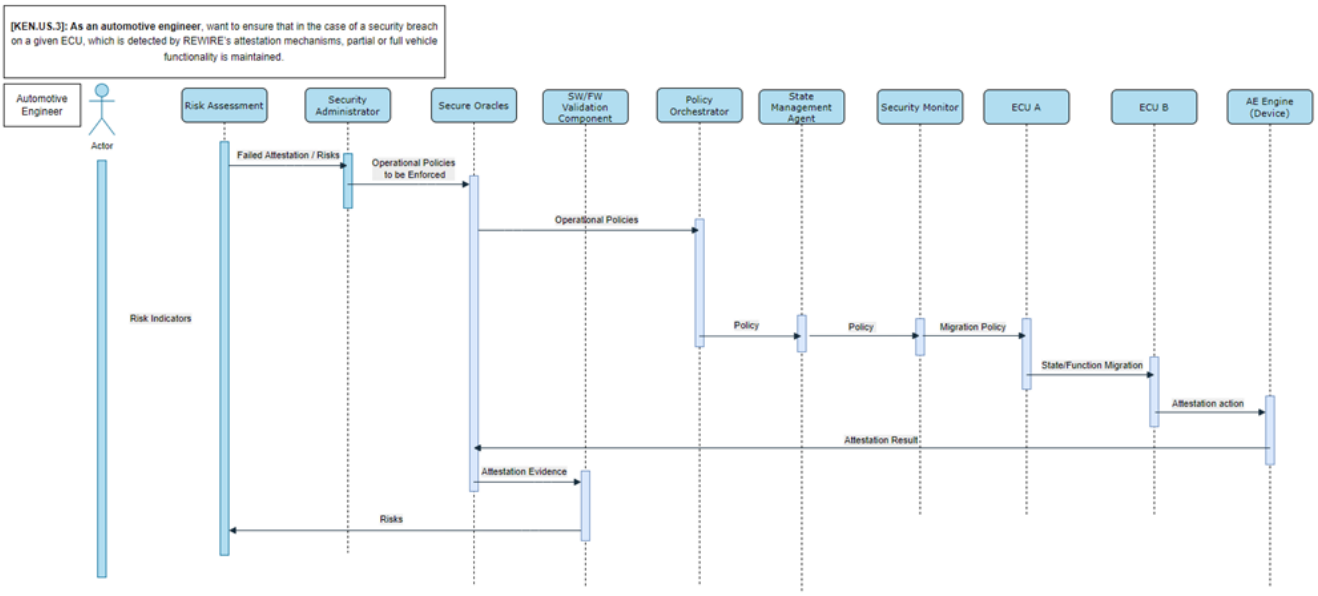

Figure 6.18: KEN US 3 Sequence Diagram

[KEN.US.4]: As an OEM system administrator I want to be able to ensure that only vehicles which are in a valid state can join and operate in a collaborative manner in the infrastructure (e.g., OEM fleet network, government-regulated road network, etc). Undisputable evidence needs to be provided for the secure state, while trust-aware authorisation needs to be performed in order to be able to exchange data in the context of the collaborative infrastructure. External entities should be in a position to verify the evidence for certification purposes.

User Story Confirmations: When a vehicle, external to the collaborative vehicle network, attempts connection to it, it should be attested for having a correct state before being able to exchange any information with the administrator or other vehicles connected to the network. Zero Trust On-boarding (ZTO) mechanisms of REWIRE will be exploited in order to regulate the process of a vehicle being onboarded to the network. In addition, all the evidence that advocate the security state of a vehicle will be logged in the blockchain infrastructure in order to enable the certification of safety critical operations.

REWIRE Functionalities:

- The ZTO mechanism of REWIRE will manage the onboarding process of a vehicle in the network.
- Security claims on a vehicle's security state will be generated in the form of verifiable presentations.
- Data collection and interfacing with the blockchain infrastructure will be facilitated by the Secure Oracles
- The REWIRE Attestation schemes will be exploited in order to take measurements of the security state of critical components of the vehicle.
- Use of predicates over verifiable attributes will be explored in the context of this use case.

The following diagram illustrates the workflows which will take place among the various REWIRE components in order to realise the user story. The diagram is based on the components as those have been documented in the main REWIRE architectural diagram in chapter 4. It is expected that the diagram and the respective interactions will be updated as the project progresses.

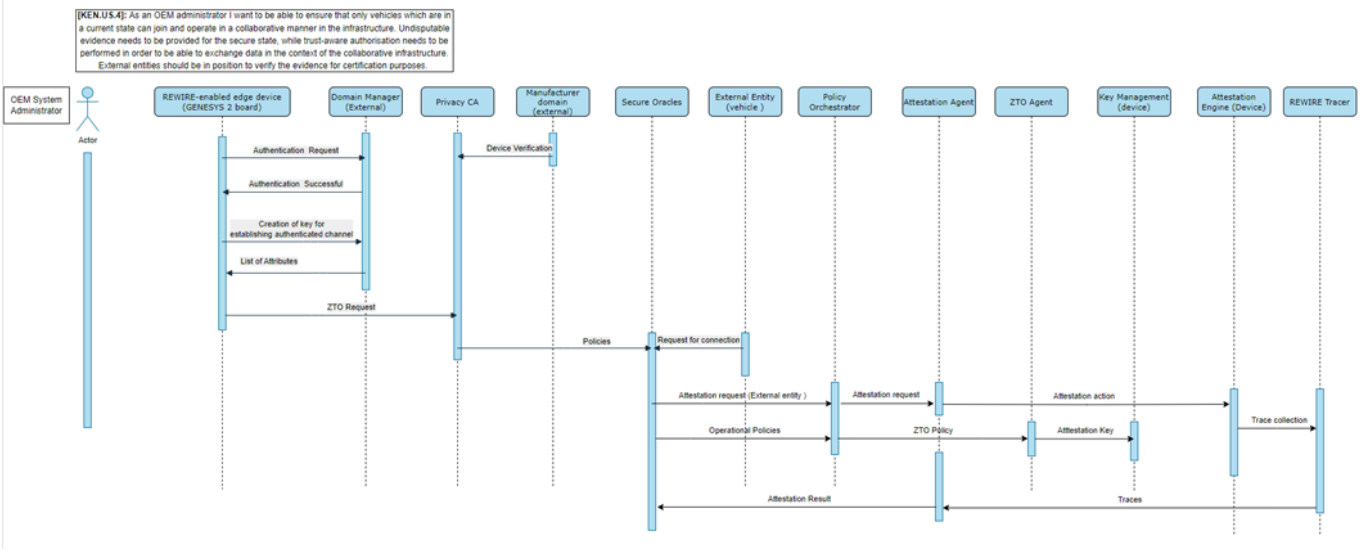

Figure 6.19: KEN US4 Sequence Diagram

6.3.5. Metrics of success

6.3.5.1. Quantitative Metrics

ID	Metric	Target Value Without TEE	Target Value With TEE	(M)andatory / (G)ood to Have / (O)ptional
1	**Time needed to perform Zero Trust On-boarding (ZTO)** (with and without the consideration of the REWIRE security mechanisms)	< 10 sec	<20% overhead	M
2	**SW update process (1-to-1 mode)** (Measuring the time for REWIRE security mechanisms (e.g., device authentication, access control), but excluding networking overhead factors)	< 200 ms	<30% overhead	M
3	**In-vehicle platform selection for migration** (including the overhead of attestation prior the migration)	N/A	<150 ms	M
4	**Critical and non-critical function migration**	N/A	True	M
5	**Disabling old version of an application**	N/A	True	M
6	**Critical function downtime during migration**	N/A	<1 sec	M
7	**Function successful migration evidence generation** (extract evidence and notify OEM)	N/A	< 10 secs	M
8	**Security lifecycle management time** (Including: detection of a device	<5 sec	<10% overhead	

	compromise, creation of VCs, alerting administrator)			

*Some target values are hard to be set a priori, since they are dependent on various application parameters, such as available computational resources, size of the SW package, number of compromised components, size of migrating functionalities, connection speeds etc. The KPIs may be refined in later stages of the project.

6.3.5.2. Qualitative Metrics

ID	Metric	Target Value	(M)andatory / (G)ood to Have / (O)ptional
1	Trusted communication channels between the cloud backend and the vehicles	To be supported	M
2	REWIRE Usability and integration	To be supported	M
3	Integrity protection of ECU configuration and behaviour	To be supported	M

6.4 Use Case – 3: Smart Satellites Secure SW Updates for Spacecraft Applications & Services

Up until now, each satellite had a specific task and was equipped with dedicated hardware and sensors to fulfil the requirements of the mission. However, it seems that this is about to change. With the increased capabilities of modern satellites and their hardware, the notion of the satellite as a service is starting to be realized. In this scheme, a satellite carries various sensors and extra hardware that can be used by various customers on demand. Instead of launching a dedicated satellite for a specific mission, customers can acquire processing time and access to specific sensors on an already orbiting satellite. This minimizes development costs and time as well as reducing the risks that commonly come along with a satellite mission (e.g., launch failure, deployment failure, failure during in orbit, etc). However, this space access model raises some serious challenges too. The satellite processing platform should ensure that the applications from various customers running on-board do not jeopardize the well-being and the availability of the satellite (e.g., misuse of the communication interfaces, excess power consumption, etc.), the mission itself and/or other applications that running on board. Security issues are also a concern as data of applications may be confidential and should not be accessed or corrupted by other applications running concurrently.

The majority of the satellites, especially CubeSats operate in Low Earth Orbit (LEO). This means that a ground station has only a limited time window to communicate with the satellite. Depending on the orbit altitude, the frequency and the capabilities of the ground station this time window can be in the order of few minutes. CubeSats normally are equipped with RF interfaces of limited bandwidth, resulting to a reduced data rate between the ground station and the satellite. Therefore, there are cases where a firmware update on a satellite component cannot be transferred on a single pass above a ground station. On top of that, a subsequent pass in the view of the ground station may only be available after a few days, significantly delaying the completion of the SW update procedure.

6.4.1. “As-is” Scenario

To deal with these aforementioned issues, LSF utilizes the SatNOGS, which is a network of multiple ground stations in different places across the globe, all connected through the cloud infrastructure of LSF. The architecture of the SatNOGS system is given in Figure 6.21 and enables operators to communicate with satellites through the distributed infrastructure of the ground stations. Having multiple ground stations in different places leads to an increased number of communication windows, thus increasing the availability of the satellite. During a SW update, a ground station at the end of the communication window can inform the backhaul infrastructure regarding the amount of data successfully sent to the satellite and which data remain to be sent for the finalization of the firmware update. The service that orchestrates the update can then find the next ground station that has an available communication window and send via internet the SW file that needs to be sent, as well as session information regarding the amount of data that have to be transmitted (see Figure 6.20).

Figure 6.20: SatNOGS multiple ground stations connected through the internet

Figure 6.21: SatNOGS architecture

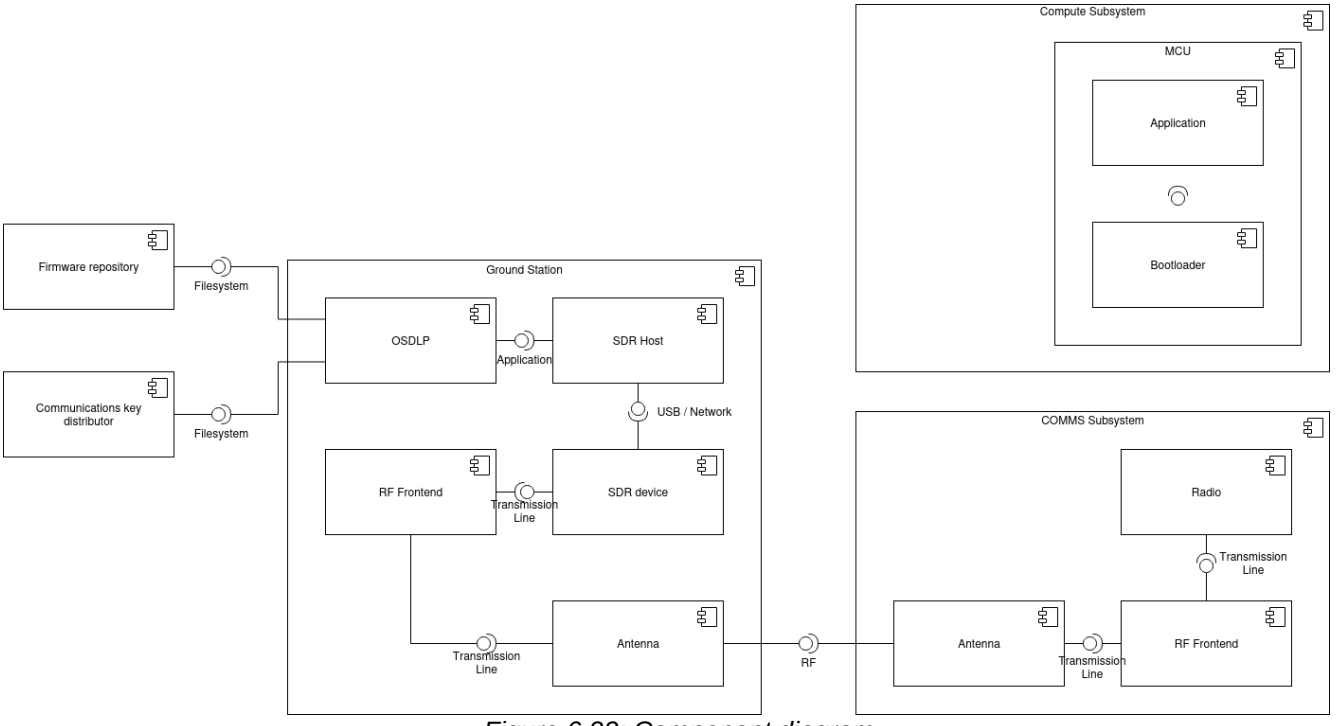

Figure 6.22: Component diagram

Currently the CubeSats and PocketCubes of LSF, operate on the Zephyr-RTOS [REF-190]. This RTOS does not provide any secure OTA functionality. To do so, LSF utilizes the MCUBoot [REF-191] bootloader, which supports asymmetrically encrypted and signed firmware images as well as fallback and multiple images management. The update process is performed by the ground station sending the appropriate telecommands. When the whole firmware is uploaded, a final telecommand instructs the firmware upgrade to take place. The MCUBoot bootloader takes over, checks for the validity of the firmware and either boots on the new firmware or falls back to the previously working one.

To handle the OTA process as a satellite passes through multiple ground stations, LSF has developed the OSDLP [REF-192], a platform and OS-independent library which implements the CCSDS Space Packet [REF-193], CCSDS TM Space Data Link Protocol [REF-194], CCSDS TC Space Data Link Protocol [REF-195], and CCSDS Communications Operation Procedure-1 [REF-196] directives. This library is responsible for the session management between the multiple subsystems of the satellite and the ground stations as the satellite passes above them. For every service and/or subsystem the OSDLP library assigns a separate virtual channel of communication, with its own session information. This session information can be either shared to the ground stations via the internet or can be retrieved by the ground station by sending the appropriate tele-command to the satellite. Normally, the first method is used because it is faster.

6.4.2. Scenario's Challenges and Needs from REWIRE

Satellites operate in a quite challenging communication environment. Physical access to the spacecraft is impossible after its deployment in space, so reliable OTA updates are the only mechanism to apply modifications or improvements to the onboard software. However, any misbehaviour during or after the OTA firmware upgrade can result to a total loss of the spacecraft. This misbehaviour can occur either from legitimate firmware or malicious one, therefore a reliable and secure firmware update procedure is of highest importance. The common challenges that satellite operators commonly face are:

- **Protection of the satellite firmware against malicious or faulty code:** Prior to the OTA procedure, the firmware files should be tested and analysed on the ground for vulnerabilities, malicious code or possible execution scenarios that can jeopardize the spacecraft and the mission in general.
- **Protection of satellite integrity against malicious code update:** Even if the firmware to be updated is guaranteed that is legitimate and error-free, the OTA procedure itself may have some flaws. An attacker could benefit from vulnerabilities in the OTA procedure, that could lead to limited availability or total loss of the spacecraft. As ground station equipment becomes cheaper, replay attacks may also be easier to implement. An attacker could re-initiate the OTA of an older problematic firmware.
- **Protection of system's integrity against Single Events Upset (SEU) and data corruption:** A common source of failures in space missions is SEU from high energy particles affecting integrated circuits and memories. A SEU can alter a legitimate firmware that is stored in the persistent or non-volatile memory, resulting to the loss of the spacecraft.
- **Fallback mechanism in case of abnormal behaviour:** In the case of abnormal behaviour of the running firmware or the OTA procedure, the satellite should fallback to a reliable and well-defined working state so a new OTA session can be re-initiated from the ground.
- **Isolation of critical operations:** Isolation is required in order to avoid different function interference which could lead to unavailability of critical services or even to satellite unavailability.

The main challenge of the current method of software updates is to verify the integrity of the firmware. Although there are methods to verify corruption of the firmware during transfer using checksums, this does not ensure that the firmware has not been tampered with. In addition to that, the communication channel is not confidential, and authentication is rudimentary. The REWIRE solution will try to cover the needs for:

- I**ntegrity against malicious code updates.** Although the binaries are protected with checksums and are asymmetrically signed, there is no secure way to transfer the firmware reliably. A malicious

actor can relatively easily apply a replay attach, initiating the OTA of an older firmware. Also, the OTA procedure itself may have security vulnerabilities that should be addressed.

- **Confidentiality and authentication of software update transfers.** Currently, the implementations of the protocols used for enabling SW updates to satellites do not cover the confidentiality and authentication requirements.
- **Trusted software update process.** Currently, there are no solution on the satellites to guarantee the integrity of SW update received. Thus, SW update validation mechanisms need to ensure that integrity of the received SW and in addition to ensure the isolated execution of the SW update process.
- **Self-healing operation.** In case of abnormal behaviour there is a need for a mechanism to recover in previously working firmware, or at a basic firmware with limited but well-defined functionality, so operators can identify the issue and re-initiate an OTA.
- **Runtime attestation of critical applications**. Critical applications are not attested during runtime. Tampering with such apps can be safety critical to a satellite. Such attestation of critical apps can be performed by REWIRE.

6.4.3. "To-be" Scenario

The satellite operator will initiate a firmware update for their satellite. The SatNOGS infrastructure will orchestrate the scheduling of the available ground stations, as the satellite passes through their field of view. Depending on the wireless link characteristics (path loss, interference level, etc.), the firmware file size and the capabilities of the satellite, the OTA procedure may not be possible to be completed into a single pass over a ground station. Each ground station informs the backbone infrastructure regarding the OTA progress. Then, based on the orbit the next available ground station is selected and informed with the OTA session information. This procedure continues until the OTA procedure is complete. During this procedure, the spacecraft should be able to mitigate possible attacks from known or unknown ground stations and provide a recovery mechanism in case of a failure during the OTA process. It will be also possible to identify firmware files that may disrupt the operation of the satellite and discard them.

In the context of this scenario, the Smart Satellites use case will be enhanced using the REWIRE technologies in to achieve:

- **Secure communication channel:** Communication channels between the ground stations in the network and the spacecraft should be secure and reliable. They should be also able adapt to the traffic characteristics, incorporating fault and delay tolerance. It also takes into consideration the limitations of the processing power of the CubeSat satellites that normally operate on microcontrollers.
- **Fail-safe and rollback mechanism:** Due to the isolated environment that satellites operate, it is impossible for a person to take action in case of a failure. Failing to apply properly and firmware update may lead to the total loss of the spacecraft. Therefore, the system should provide a fail-safe mechanism that sets the satellite in a well-known state in case of failure. The system should also provide a rollback mechanism, to boot in a previously working firmware in case the current exhibits abnormal operation. Genesis board will host the attestation artifacts.
- **Analysis of the firmware file:** Before applying the firmware on board, basic analysis of the firmware should be performed in order to detect possible vulnerabilities and/or implementation flaws so that to prevent abnormal operation
- **Attack identification and handling:** The system should be able to identify possible attacking sources onboard the spacecraft in terms of applications that may deviate from a specific behavioural profile or in terms of critical services whose binaries and configurations may have been modified.
- **Services availability during the OTA firmware update procedure:** The update procedure and all the security countermeasures should not affect the availability of the satellite and its services. Proper function isolation measures should be considered.

The components that the system will operate include:

- **SatNOGS infrastructure:** The backbone infrastructure that supports the entire SatNOGS project, orchestrating the deployed ground stations and over-viewing the available orbiting satellites. It consists of several cloud-based services and databases
- **SatNOGS ground station:** A ground station, with one or more antennas and RF interfaces and enough processing capabilities to perform DSP tasks
- **Satellite:** A satellite with one or more subsystems

6.4.4. Reference scenario user stories

[LSF.US.1]: As a satellite operator, I want to initiate a secure and reliable SW update on the satellite using the SatNOGS fleet of ground stations around the globe. The SW update may be accomplished by more than one ground station, based on the trajectory of the orbiting satellite.

User Story Confirmations: The SW update package will be checked in terms of integrity and authenticity. The SW update will be applied only if it carries the appropriate security credentials, defined by the REWIRE security framework. The communication between the mission control centre that initiates the OTA, and the satellite will be secure, preventing replay attacks. The security between the mission control and the satellite should be assumed end-to-end, allowing the satellite to receive frames from any ground station as it passes through them, avoiding complex mechanisms (e.g., on-satellite key management, regular updates of possible new GS keys, etc.), simplifying ground stations on-boarding and the satellite operations in general.

REWIRE Functionalities: To achieve a high level of security in REWIRE, there are certain needs for OTA features and procedures which will be addressed by the REWIRE offerings.

- Before launch, the operator requests security credentials for her satellite from the REWIRE infrastructure and stores them at the satellite. In this way, we will be in position to utilise the side-channel resistant mode of operation of AES designed by REWIRE.
- The mission control centre encrypts and authenticates the firmware file issued by the satellite operator. This requires that the SatNOGS mission control centre can access the REWIRE security components
- Attestation method is performed after the satellite fully receives the new firmware file so that to test the integrity of the new state of the satellite.
- If any attestation mechanism fails, the satellite makes this information available on the periodic telemetry frames. With this way, subsequent ground stations will receive this information and report it back to the mission control centre. This increases the awareness of the satellite operator for the failure in the OTA process or possible attack, utilising also the risk assessment component of REWIRE.
- If the uploaded SW is malicious or corrupted, the satellite rolls-back to the previous working firmware. (More details are given in LSF.US.5)

The following diagram illustrates the workflows which will take place among the various REWIRE components in order to realise the user story. The diagram is based on the components as those have been documented in the main REWIRE architectural diagram in Chapter 4. It is expected that the diagram and the respective interactions will be updated as the project progresses.

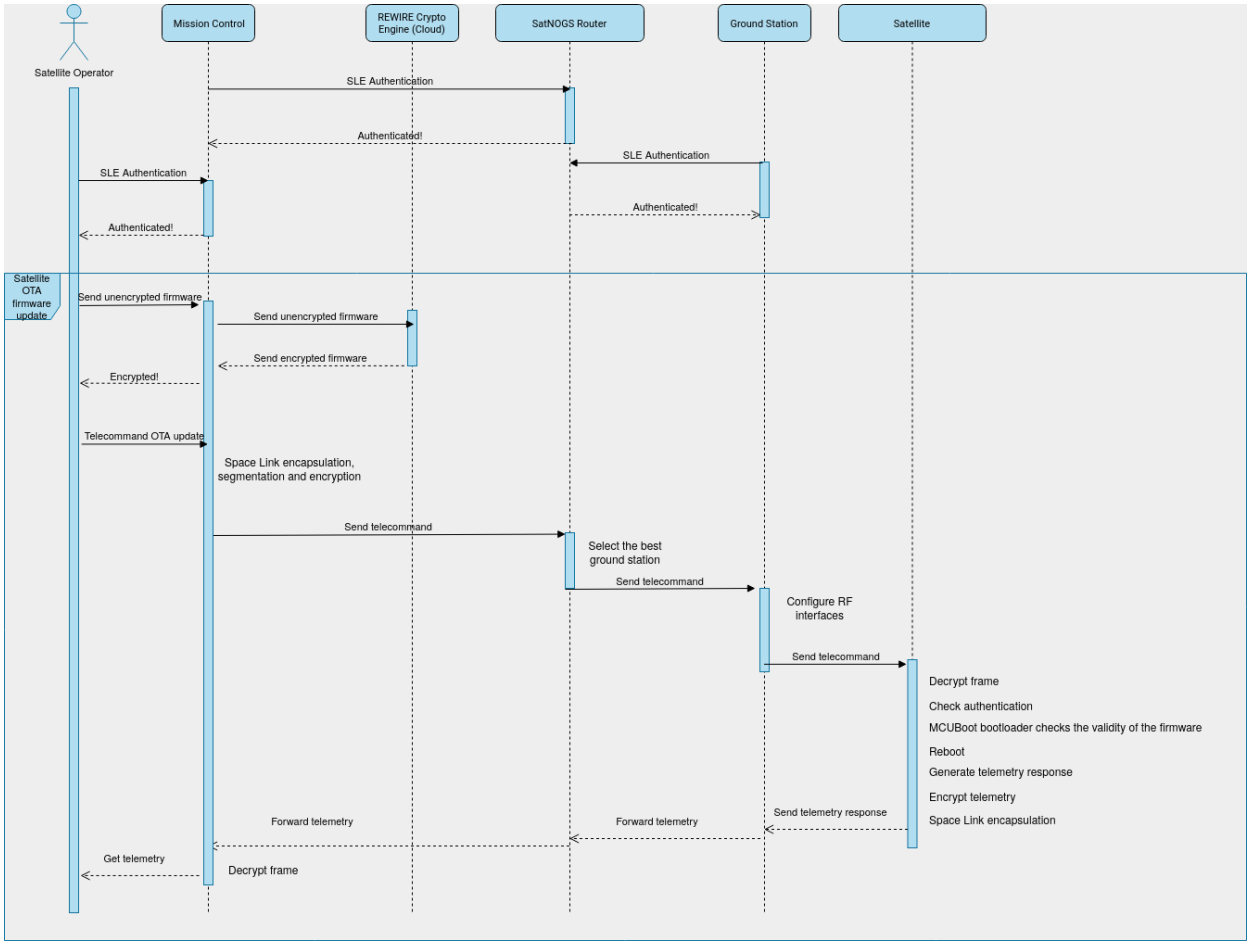

Figure 6.23: LSF US 1 Sequence Diagram

[LSF.US.2]: As an operator of a network of connected ground stations, I want new ground stations to be able to join the network in a secure and authenticated way, with minimal human interaction. The status of the ground station should be easily accessible by the SatNOGS mission control centre. **As a ground station operator,** I would like a ground station to join the network with the minimal amount of configuration procedures.

User Story Confirmations: Only authorized and authenticated stations should join the network of ground stations. In addition, a secure channel should be established with each of the ground stations and the SatNOGS Router entity, which is responsible for the orchestrion of the entire fleet of ground stations. Even if the model of communication between the mission control centre and a satellite utilizes end-to-end encryption for most of the missions, ground stations and the SatNOGS Router exchange sensitive data like for example the exact location of the ground station, owner information, etc. Therefore, it is necessary that a secure channel communication between the SatNOGS Router and each ground station is established. The system should also be able to revoke any security credentials in the case that a ground station is compromised or misbehaves so it does not pose a threat to the entire network.

REWIRE Functionalities: To achieve the secure and authenticated on-boarding of ground station, REWIRE should:

- Use the ZTO protocols in order to onboard in a dynamic manner the various ground stations that ensure the stable communication channel with the satellites.
- During on boarding REWIRE would validate also the operational correctness of the ground stations. To do so, VCs will be used to bear the attestation result as evidence/security claims

The following diagram illustrates the workflows which will take place among the various REWIRE components in order to realise the user story. The diagram is based on the components as those have

been documented in the main REWIRE architectural diagram in Chapter 4. It is expected that the diagram and the respective interactions will be updated as the project progresses.

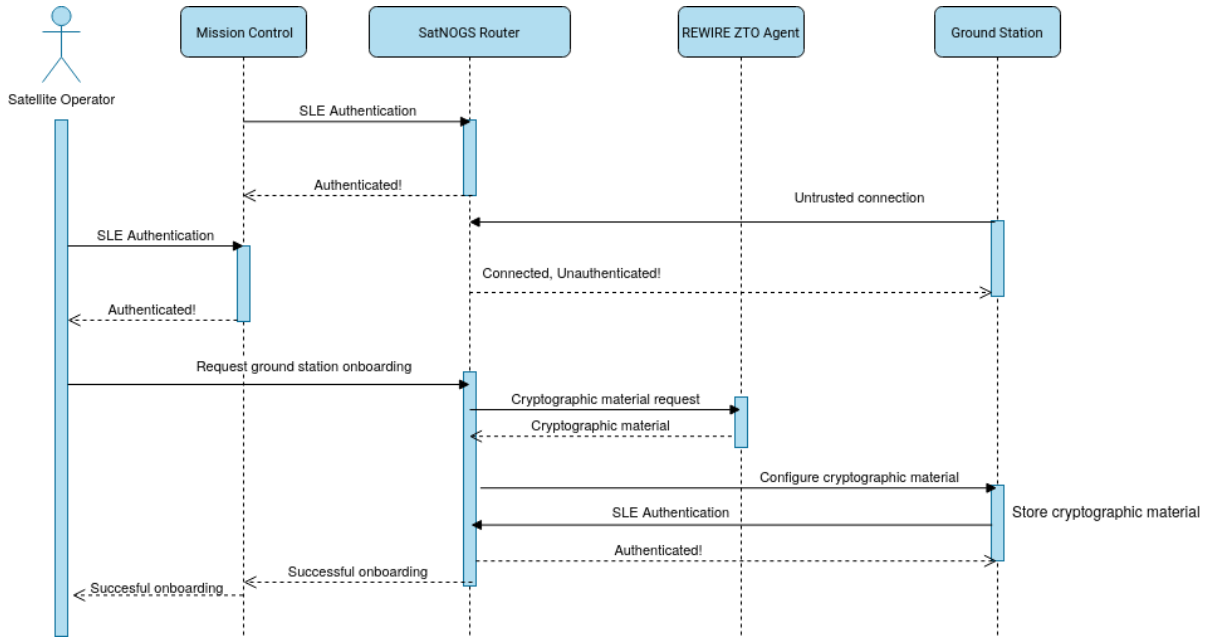

Figure 6.24: LSF US 2 Sequence Diagram

[LSF.US.3]: As a satellite operator, I want to automatically identify a misbehaviour on my satellite, based on the periodic health telemetry data that are fetched automatically by the SatNOGS network of ground stations.

User Story Confirmations: Satellites send periodically a lot of information regarding the health of the spacecraft including but not limited the battery status, temperature readings, the charging status, the pointing and position of the spacecraft relatively to earth and sun, current flowing on different power rails and subsystems. In addition, they might also include the status of the operating system and the on-board software, the CPU and memory utilization as well errors that may have been encountered. On-board data fusion for misbehaviour detection may require a significant amount of processing power and may even be impossible due to vendor-locked on-board subsystems. The amount of historic data may also be a limited factor too. Therefore, it is more convenient to perform such an analysis on-ground, by automatically fetching the telemetry data from the SatNOGS infrastructure, that allows full access to all the historic telemetry data. An AI system can gather and analyse these data on regular intervals and automatically inform the satellite operator in case a misbehaviour is detected (e.g., no charging even if the temperature is within acceptable levels and the satellite solar panels facing the sun).

REWIRE Functionalities: For such a system the following REWIRE functionalities need to be combined:

- Facilitate data retrieval from the SatNOGS database of decoded frames for a specific mission. The use of secure oracles will be explored as the prominent way to achieve this.
- REWIRE should take advantage of the AI-based misbehaviour component so that to train the AI model based on the available data
- REWIRE should automatically and periodically execute the AI system to spot misbehaviours
- Upon a misbehaviour detection the REWIRE AI system should inform the SatNOGS mission control centre and the risk assessment component.

The following diagram illustrates the workflows which will take place among the various REWIRE components in order to realise the user story. The diagram is based on the components as those have been documented in the main REWIRE architectural diagram in Chapter 4. It is expected that the diagram and the respective interactions will be updated as the project progresses.

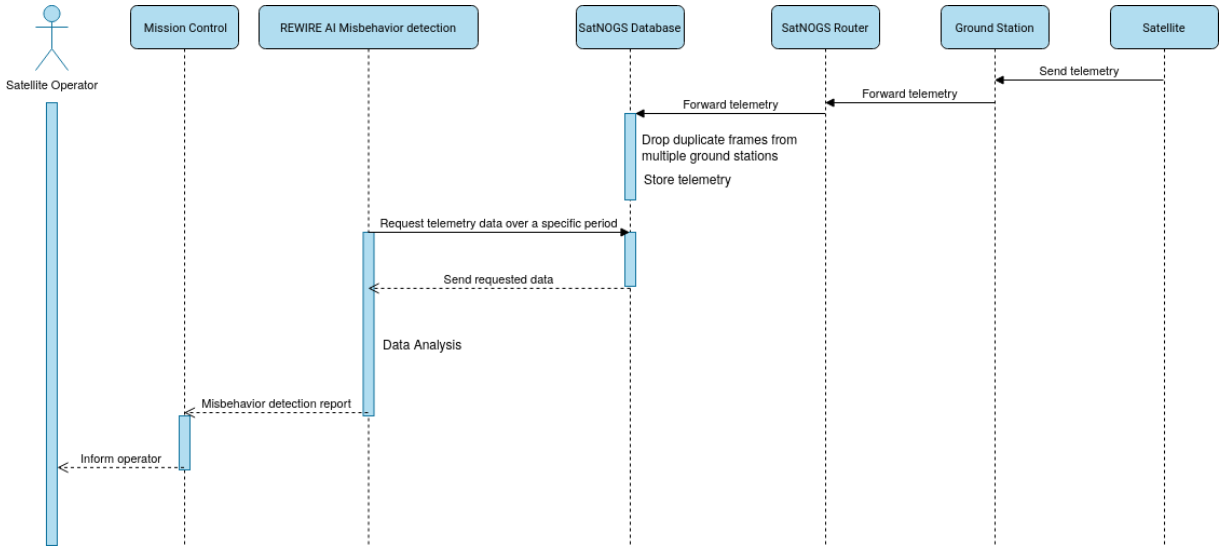

Figure 6.25: LSF US 3 Sequence Diagram

[LSF.US.4]: As a satellite operator, I would like the satellite on-board software to have a fallback mechanism in case a misbehaviour is identified. The fallback mechanism should be reliable and autonomous, as during the deployment in space no human intervention is possible.

User Story Confirmations: Satellites operate in a quite harsh environment. The major challenge is the total lack of physical access, with the only options left for correction actions to be either remotely controlled or autonomously on the spacecraft. Errors that can occur on the onboard firmware can be both internal (e.g., software bugs, deadlocks, unhandled erroneous states, etc.) or extraneous (e.g., bit flips or faulty memory banks caused by radiation). Regardless of the root cause, such issues may lead to the total loss of the spacecraft if they are not identified rapidly and handled properly. Therefore, it is essential for the satellite to identify a misbehaving firmware and switch immediately the satellite into a fallback mechanism, allowing the operators to identify the root cause and take evasive actions. The fallback mechanism may also try to select previously working firmware files stored onboard, in an effort to minimize the satellite service downtime.

REWIRE Functionalities: For such a system REWIRE should provide:

- Continuous device state monitoring during runtime, utilising the attestation mechanisms of REWIRE.
- Attestation mechanism for misbehaviour identification during runtime and control flow attestation.
- The REWIRE TEE in implement the fallback mechanism

The following diagram illustrates the workflows which will take place among the various REWIRE components in order to realise the user story. The diagram is based on the components as those have been documented in the main REWIRE architectural diagram in Chapter 4. It is expected that the diagram and the respective interactions will be updated as the project progresses.

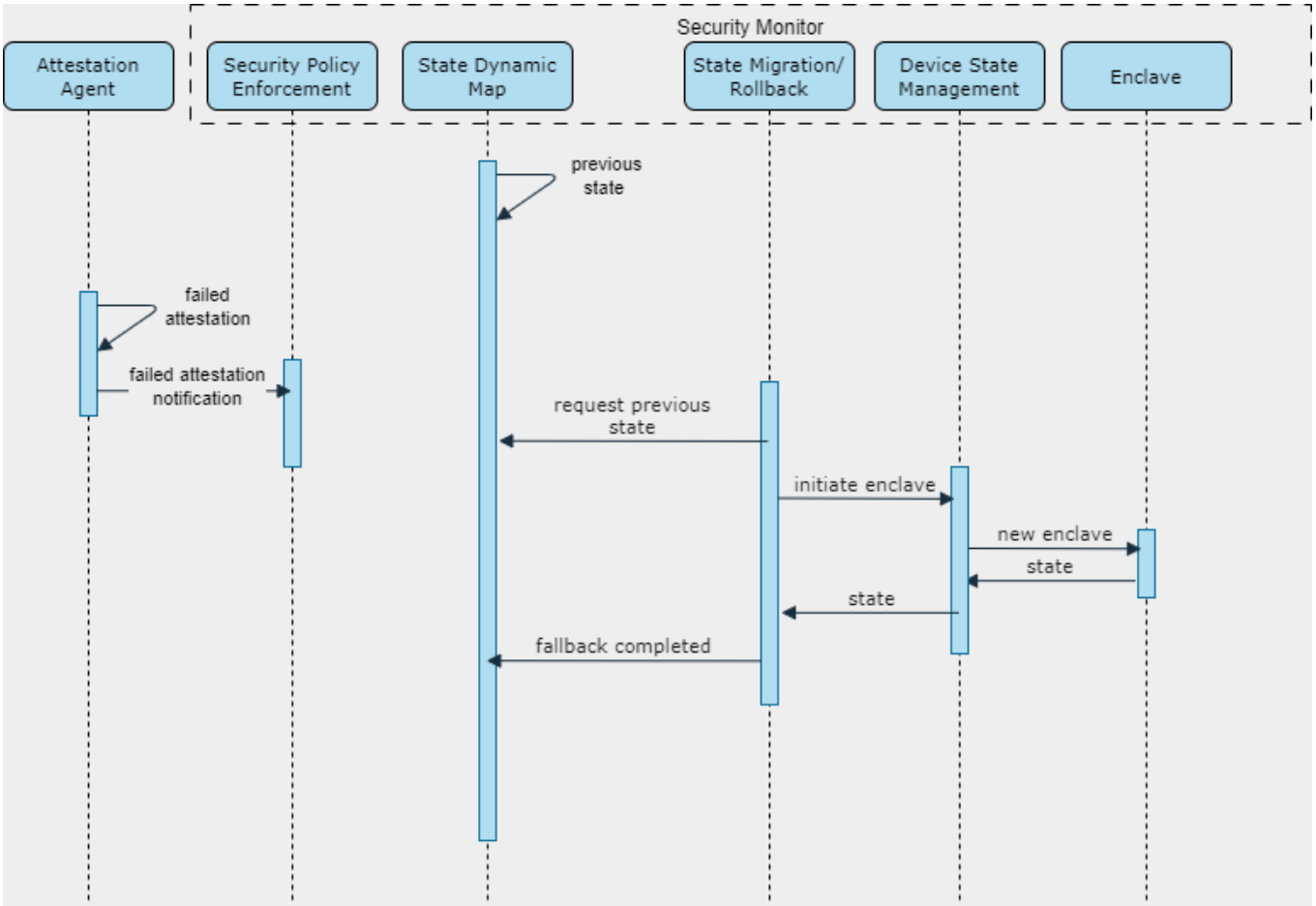

Figure 6.26: LSF US 4 Sequence Diagram

6.4.5. Metrics of success

6.4.5.1. Quantitative Metrics

ID	Metric	Target Value	Acceptance Value	(M)andatory / (G)ood to Have / (O)ptional
1	**Power consumption** (REWIRE artifacts will be evaluated individually)	< 10% overhead	30% Overhead	M
2	**CPU load** (REWIRE artifacts will be evaluated individually)	< 10% overhead	30% Overhead	M
3	**Duration of firmware update**	< 12 mins	-	G
4	**False positive of AI misbehaviour detection**	< 5%	-	G
5	**False negative of AI misbehaviour detection**	< 2%	-	G
6	**Time of a ground station onboarding**	< 1 sec	-	M
7	**Satellite downtime during the OTA completion**	< 30 secs	-	M

8	**Fallback firmware procedure duration**	< 120 secs	-	M

6.4.5.2. Qualitative Metrics

ID	Metric	Target Value	(M)andatory / (G)ood to Have / (O)ptional
1	**Satellite operator satisfaction**	High	G
2	**Configuration steps required for a ground station onboarding**	Minimal	M
3	**Multiple fallback firmware files support**	Yes	G

Chapter 7
Trust indicators and threat landscape analysis

This chapter explores the trust indicators and properties related to the REWIRE framework and the operational landscapes where it will be applied. Specifically, we review the required level of evidence to prove the trustworthiness of operations and services run on IoT devices, whilst taking into consideration adversarial assumptions and an attackers' capabilities. We need to highlight that this Chapter is a high-level introduction to REWIRE threat landscape categories. More detailed definitions and analysis of the discussed threats will be performed in the future WP2 deliverables (D2.2), as well as the individual technical deliverables of the REWIRE components (WP3-WP5), whereby T3.4 undertakes the automated risk assessment methodology.

7.1 Trust Establishment in the REWIRE Ecosystem

The REWIRE trust framework will be used to formally describe the policies, procedures, and mechanisms for the operation of digital trust in the network. Informally, trust and trustworthiness must capture a varied set of relationships between entities, hosts, and other components of the network. For instance, between two users in the system, one user and a component, or two components. In the following, we provide high-level discussion regarding the trust relationship between two entities, metrics of trust, the importance of zero-trust in REWIRE, and the concept of trustworthiness.

Trust Relations:
Incorporating a trust assessment mechanism, in addition to cryptographic mechanisms, is imperative given REWIRE's complex, heterogeneous ecosystem of devices and mixed-criticality services. For instance, IoT devices are inherently untrustworthy, thus, a trust assessment mechanism plays an important role in measuring the level of trust between entities and components in the system.

More formally, trust depicts a ternary relation in which entity A expects and can rely on entity B (denoted $T_{A,B}$) for some outcome X. The outcome is known before the trust relation is established. That is, entity B states their intentions to perform goal X which is usually a desired outcome for entity A. Note that entity A does not need to value the outcome; reliance is sufficient.

Entity A places expectations (R) on the performance, and trust may need to cover the how (process), the why (purpose), and sometimes the context (C) of the goal. Formally, trust can be captured by the 3P's (Performance, Purpose, Process) [REF-197] and we can write $T_{A,B} \Longrightarrow R(X)$ in C. Moreover, trust can be measured to an extent or so-called degree (D) based on some evidence (E).

This can be written as $T_{A,B} \Rightarrow R(X)$ in C to degree D given evidence E.

Evidence provided to entity A of entity B's behaviour can be direct or indirect. Further, time may be incorporated such that evidence expires or the context is no longer stable after some time (T).

Generally, an agent will perform an assessment of whether entity B can exhibit $R(X)$, formally known as a referral of trust.

Trust relations can be categorized into three types:
1. A trusts B.

2. A has limited trust in B.
3. A distrusts B (A has negative expectations of B demonstrating $R(X)$ in the context of C).

Moreover, trust is directional/asymmetric and currently captured between two entities. Theoretically, if we extend binary trust to n entities, there are n*(n-1) possible relationships. Furthermore, a trust relation involving more than two entities implies a collection of entities is required to demonstrate $R(X)$ behaviour. Ultimately, this means that an entity can form a trust relation with the whole system rather than a component of the system.

Metrics and Anchors of Trust:

As mentioned earlier, a trust assessment mechanism will play an important role in REWIRE to quantify the trust between entities and components that inherently lack trust. Measuring trust can be done in several ways:

1. A continuous/discrete value represented in the interval [-1,1] such that the minimum value represents distrust, and the maximum value represents complete trust.
2. Alternatively, assign values to $(a, b, c)\epsilon[0,1]$ representing trust, disbelief, and uncertainty respectively. Then the following holds: $a + b + c = 1$.
3. Use a probability distribution whereby the mean value represents a trust value for instance.

We also need to capture the root(s)-of-trust of security properties to enable trust in components of the system. Note that some HW may be deemed secure enough to be completely trusted. The following concepts have been established as anchors of trust [REF-198]:

1. Legal regulations recognised by jurisdictional legislation and/or contractual agreements. This anchor of trust underpins operational processes and rules within the trust framework. Formally, legal trust anchors are used to incentivise (disincentivise) entity B to demonstrate (deviate from respectively) $R(X)$.
2. Data sources relating to entities and attributes to be processed.
3. Cryptographic root-of-trust for ensuring the properties of binding, revocation, authentication, signing, encryption, and other trust functions.
4. Cybersecurity anchors such as policy violations/enforcement, certification processes etc.

Zero Trust:

The dynamic, heterogeneous nature of the REWIRE ecosystem calls for the integration of a trust paradigm throughout the network and communication processes. In the context of similar environments, it has become increasingly popular to adopt the zero-trust stance, which informally means actively verifying processes, devices, and components of devices in runtime.

In more detail, zero-trust security is a security model requiring strict identity verification for every subject/entity and device trying to access resources on a private network, whether they are inside or outside the perimeter of the network. That is, a ZTA [REF-199] follows principles which assume no implicit trust granted to assets or user accounts based solely on their physical/network location or asset ownership. Adherence to the following tenets is required in a ZTA:

- All data sources and computing services are considered resources.
- All communication is secured regardless of network location.
- Access to individual enterprise resources is granted on a per-session basis.
- Access to resources is determined by dynamic policy—including the observable state of client identity, application/service, and the requesting asset—and may include other behavioural and environmental attributes.
- The enterprise monitors and measures the integrity and security posture of all owned and associated assets.
- All resource authentication and authorization are dynamic and strictly enforced before access is allowed.
- The enterprise collects as much information as possible about the current state of assets,

network infrastructure and communications and uses it to improve its security posture.

Trustworthiness:
Trustworthiness is the complement to trust in the assessment framework, whereby trustworthiness in REWIRE is predicated upon the trustee entity/component being deemed worthy of trust.

Rather than being defined by entity A's expectations of entity B, trustworthiness is a property of B. Specifically, it is a measure of B's ability to fulfil the role/actions relied upon by A. Broadly, we can say that entity B is trustworthy for entity A if A can rely on B. Modifying the notation of previous equations, we can depict trustworthiness as $TW_{B,A} \Rightarrow R(X)$ in context C given evidence E. Note that evidence of entity B's ability leaves no room for uncertainty, thus, it is a guarantee of trustworthiness.

Ideally, some entity Z can measure and observe $TW_{B,A}$ (B is trustworthy for A) then provide an assessment to A. Given entity Z is independent, the assessment must consider entity A's expectations and requirements. In the context of trust relations, if there is a lack of evidence of $TW_{B,A}$ then entity A should adopt the limited/dis-trust stance regarding entity B (trust type 2 or 3).

ISO/IEC TS 5723:2022 defined trustworthiness as the ability to meet stakeholders' expectations in a verifiable way.

7.2 Adversarial Model and Assumptions

In this subsection, we focus on analysing an adversaries' capabilities in the REWIRE ecosystem, categorising attacks into three types. Namely, host-based, network-based, and physical/algorithmic-based attacks. In doing so, we can examine threats to security, identify appropriate security mechanisms, and explore gaps in the literature that REWIRE can potentially address.

Intel SGX was shown to be vulnerable to side-channel attacks based on cache [REF-200], page-faults [REF-201] as well as so called speculative execution attacks [REF-202]. Keystone, through the Security Monitor, can guarantee the confidentiality and integrity of the contents inside an enclave but can also be vulnerable under several threats. After the boot process, a measurement of the Security Monitor is generated and signed by the trusted hardware. If this signed measurement corresponds to the expected version, the Security Monitor is trusted. The Security Monitor only trusts the hardware, while the host and the runtime (RT) both trust the Security Monitor. The enclave app (eapp) trusts the Security Monitor and the RT. The approach suggested in [REF-203][REF-204] for a security analysis of Keystone is based on the protection of the enclave, the host OS and the Security Monitor. Only the Security Monitor and the enclave have PMP access to that enclave's contents (these including not only the application but also the RT). This, in addition to the attestation primitive offered by Keystone, which enforces integrity at the initial state of the enclave, complicates software attacks to the enclave.

Physical, software, side-channel, and denial-of-service attackers [REF-203] are listed as potentially willing to compromise the security guarantees of the system. Some of them are more relevant to the REWIRE scenarios than others, and even some of them are more relevant for a particular device inside a given scenario (e.g., a physical attacker may not be of substantial relevance for a physically protected device on the cloud plane). These will be detailed in the ensuing table. Physical attacks are considered by creating an *on-chip memory extension* [REF-203] such that it is possible to protect the enclave from a physical attacker A_{Phy} who has access to the DRAM. The purpose of this is to enable the enclave to execute without code or data leaving the chip package. This can be done – depending on the underlying platform – by dynamically instantiating a scratchpad memory region using a memory controller that can be used, exclusively, by said enclave for its entire lifetime. It is not a major modification and does not change the RT or the eapp.

To protect the enclave from a cache side channel attacker (A_{Cache}), a cache partitioning scheme can be implemented [REF-203] utilizing hardware capabilities and again PMP. This enforces non-interference between the enclave and the untrusted world during the enclave execution, since then only the cache

lines from the enclave physical memory are in the partition and are thus protected by PMP. Also, the adversary cannot insert cache lines in this partition during the enclave execution due to a line replacement waymasking mechanism. Other additional functionalities, such as page encryption – in the fashion of Intel SGX – can be added to a modular RT like Eyrie. In the event of a page fault, an enclave might try to evict pages outside the secure physical memory. Page encryption and integrity protection can be implemented in order to maintain confidentiality (and integrity) of the page contents when moving it to insecure storage. This could also be used as a feature, although it implies important design challenges. Side-channel attacks (A_{SC}, A_{Time}) are commented in the "Protection of the Security Monitor" paragraph.

Protection of the host OS: Attacks from an enclave to the host OS are not possible due to the own limitations by which an enclave operates. For instance, it cannot edit a page table that does not belong to the enclave and has no reference of other memory regions outside its boundaries. The Security Monitor timer can help prevent an eapp trying to DoS the system by simply interrupting the enclave execution and return execution to the OS. Furthermore [REF-204], the enclave cannot corrupt the host state because the Security Monitor performs a full context switch when the execution passes from enclave to host OS and back. However, Keystone currently has no meaningful mechanisms to protect against speculative execution attacks [REF-205][REF-206]. These are a combination of A_{SW} and A_{SC} by inducing a victim to speculatively perform operations that would not occur during correct program execution, and which leak the victim's confidential information via a side channel to the adversary. Defences against these attacks can nevertheless be implemented in Keystone [REF-207][REF-208].

Protection of the Security Monitor: The Security Monitor, after boot, reserves and occupies a certain memory region that is PMP protected from the host OS and the enclaves, making it impossible for an A_{SW} to attack it. However, it is possible for the RT to invoke untrusted system calls into the OS. In these cases, there is a risk of Iago attacks via the untrusted interface, whose protection is delegated to the RT and eapp developer. Mechanisms for such protection can be implemented [REF-209][REF-210]. One typical origin of such an attack is using legacy code as source for the eapp, by porting a legacy application. Here, while the TEE provides isolation for the application's runtime state and memory, the system call interface represents a significant attack surface for the trusted application. For code to be vulnerable, it must a) neglect to sanitize the return values of a system call and b) use the return values in an unsafe way. A more general definition for these attacks is *syscall tampering attacks*. Regardless the origin, Keystone can reuse existing shielding systems [REF-211] as RT modules to defend against these attacks – however it is recommended to include only strictly necessary features in the RT and thus reduce the TCB.

The Security Monitor SBI is vulnerable since there is no non-interference guarantee for it. However, the SBI in Keystone is a well-defined and well-limited interface from the Security Monitor to the S-mode (RT and host OS): it does not perform complex resource management and together with the low code footprint of the Security Monitor make the latter able to be formally verified, according to [REF-212]. The more extensions added to the SM SBI – and thus the more capabilities added to the Security Monitor – the more difficult for the SM to be formally verified and the more vulnerable the SBI becomes. Natively, Keystone does not protect the Security Monitor (or the enclaves) against timing side-channel attacks. However, software solutions exist for masking timing channels [REF-213] and hardware manufacturers can supply timing attack resistant hardware (e.g., Rocket-based quad-core SoC chip (FU540) with a proprietary L2 controller as in [REF-203]). Designers must however avoid naive deployment of Keystone (or similar TEEs) on out-of-order processors such as BOOM and XiangShan, which cannot guarantee isolation [REF-214]. To protect the Security Monitor against a physical attacker A_{Phy} the security monitor should be executed entirely from the on-chip memory. The Security Monitor is statically sized and has a relatively small in-memory footprint (< 150Kb). On the FU540, this would involve [REF-203] repurposing a portion of the L2 loosely integrated memory (LIM) via a modified trusted bootloader.

In the context of REWIRE and TEE Keystone, a physical attacker A_{Phy} can intercept, modify or replay signals that leave the chip package if it has physical access to it. Here, A_{PhyC} is for confidentiality, A_{PhyI} is for integrity. Moreover, a side-channel attacker A_{SC} can obtain information by observing interactions between the trusted and the untrusted components via the cache side channel (A_{Cache}) or the timing side channel (A_{Time}). For instance, cryptographic keys or other information could be revealed by analysing execution time patterns. Controlled channel attacks are not included since enclaves do not share any

state with the host OS or the user application. In the following Table 7.1, we categorise types of attacks and an attacker's capabilities into physical, network-based, and host-based attacks.

Table 7.1: Attacker Capabilities and Types of Attacks Categorisation

Attackers' Capabilities	Threats to Security	Identified Security Mechanisms	Gaps in the Literature
Run SCA against specific device to recover the long-term key (Physical Attack)	Confidentiality and integrity violation	SCA-secure mode of Authenticated Encryption	Often heuristic solution No clear definition of security in the presence of leakage Special skill needed for secure implementation
Run SCA on a device in order to inject a single fake message (without recovering the long-term secret) (Physical Attack))	Integrity violation	SCA-secure mode of Authenticated Encryption	Often heuristic solution No clear definition of security in the presence of leakage Special skills needed for secure implementation
Run SCA on a device in order to read a single message (without recovering the long-term secret) (Physical Attack)	Confidentiality violation	Out-of-scope: the cost vs. benefit of preventing this attack is too high. From defender's perspective, preventing an SCA attacker from reading data involves protecting, not only the decryption itself, but also the full SW update process afterwards. From attacker's perspective, running a full SCA in order to read the content of one specific SW update, without the possibility to extend this to further updates, probably represents disproportionate effort.	
A software attacker A_{SW} can control host applications or the untrusted OS, can introduce malicious information in network communications, launch adversarial enclaves, arbitrarily modify any memory not protected by the TEE, and add, drop or replay enclave messages.	Authenticity, integrity, and confidentiality violations	Cryptographic mechanism to ensure secure SW update. The main requirement is to ensure that the update is transferred in a secure manner.	The cryptographic solution for provably secure SW updates needs to be easily deployable to a large class of IoT platforms, as in the REWIRE architecture.

(Host-Based Attack)			
Attacker can eavesdrop the communication channel or try to inject fake messages (Interception Attack). Thus, adversary can access information without authorisation. (Network-Based Attack)	Privacy, Confidentiality, and Integrity Violations	Authenticated Encryption (AE)	No gaps. AE is a well-understood solution
Sybil Attacker uses a single node in the blockchain to operate many false identities simultaneously (Network-Based Attack)	Authorisation and confidentiality violation	Identity verification mechanism and social trust graphs analysing data connectivity	Identity validation (direct/indirect) sacrifices anonymity – work on proof of personhood solutions to provide anonymity. Over-reliance of SSI can lead to challenges: false identities in the verifier application, issues inserting and defining VCs held in digital wallet, interoperability between system and SSI solutions.
An impersonation attacker is a malicious actor pretending to be someone else, or multiple entities, to steal sensitive data from an entity in the system. Examples include phishing attacks in which social engineering tactics are used. (Network-Based Attack)	Confidentiality violations	Identity verification mechanism and social trust graphs analysing data connectivity Runtime attestation of device/component state and ownership	An open problem is how a user can insert a runtime attestation result, in the form of an attribute, into a VC issued from a VC (or DAA) issuer.
Man-in-the-Middle is a type of eavesdropping attack in which an attacker puts themselves in the middle of two entities, typically a user and an application, to intercept their communications and data exchanges and use	Authorisation problems leading to privacy, confidentiality, and integrity violations	Authenticated Encryption (AE)	No gaps. AE is a well-understood solution

them for malicious, unauthorised actions. (Network-Based Attack)			
Denial-of-Service attacker can take down the enclave or the host OS. Keystone allows the OS to DoS enclaves as the Security Monitor do not manage resources while the OS can refuse services to user applications at any time. (Network-Based Attack)	Availability violations	Network segmentation and implementing a zero-trust paradigm	The heterogeneous nature of systems like REWIRE and the evolving threat landscape makes it challenging to implement secure solutions. Interoperability issues between IoT devices and software-defined network vendors.

Chapter 8

Security Lifecycle Management and Secure On-Boarding and Monitoring

In this Chapter we focus on the dynamic nature of the REWIRE ecosystem and the mechanisms in place to monitor the whole architecture to manage the security of the system throughout its lifecycle. In particular, we provide a high-level introduction to key management systems incorporating key hierarchies with which a variety of keys used in the REWIRE architecture are derived. Moreover, we recall Section 4.4.7, regarding secure device onboarding, by detailing the creation of key restriction usage policies which are utilised by access control mechanisms.

8.1 Key management and key hierarchies

REWIRE requires the design of Key Management System (KMS) which can operate even in an (un)-trusted environment. Additionally, REWIRE requires a scalable solution with respect to the number of devices engaged, with enhanced performance regarding the time to derive cryptographic keys. Given these goals, we consider a Keystone TEE enabled KMS.

At a high level, access control mechanisms are used to provide only authorised users with access to data. Data is encrypted to provide confidentiality and related files/data are encrypted by a cryptographic key. Therefore, storing cryptographic keys securely in a KMS is essential to support the distribution of keys to authorised users. It is desirable to *minimise* the amount of plaintext keying material requiring protection and storage. One approach is to use a key management hierarchy [REF-216], such that which is beneficial to reducing storage of keys, segmentation of encrypted data, and the design of appropriate key-restriction usage policies.

A hierarchy is an *acyclic graph* of keys assigned to nodes. A data file is associated with a node in the graph and encrypted with the associated cryptographic key assigned to the node. Moreover, an authorised user with a node key can efficiently compute any descendant node's key in the hierarchy, however, it is assumed to be computationally infeasible to derive a non-descendant key in the hierarchy [REF-217]. A key management hierarchy can be categorised into two district types: user-based and resource-based. In the former, nodes in the graph represents a group of users' who have access to the corresponding node key. The latter type of hierarchy represents a group of resources such that an authorised user with access to the assigned node key can then access the resource(s) associated with the node.

In the context of REWIRE, a KMS will be the baseline for several requirements such as secure device onboarding, providing keys for necessary cryptographic primitives (e.g., digital signature schemes) for dynamic credential management, and achieving secure SW/FW updates. The types of keys requiring storage includes symmetric and asymmetric keys regarding runtime attestation, sealing, migration etc. We highlight that the cryptographic keys cannot be derived in a secure way regarding side-channel analysis attacks.

The exact types of keys used in the REWIRE architecture will be defined in work package deliverable D4.1 as part of the zero-touch onboarding mechanism, in which key restriction usage policies are created. The ensuing is a less detailed list of the types of keys, specific or general, that will be required in the REWIRE.

1. **Device Identity Key:** A unique, unclonable, HW-backed identity key that can be authenticated and factored into access control decisions (key restriction usage polices) during the secure enrolment process.
2. **SW/FW Update Key**: a pre-established, symmetric key used for SW updates, that is leakage resilient. Activated following the onboarding process (see section 4.4.7).

3. **Long-term Identity Key:** a storage asymmetric key-pair created within the TPM and used to protect the TPM protected keys stored outside of the TPM to provide fine-grained access control and acts as the root key of attribute signature keys in the key hierarchy. Crucially, it is used for the authentication of the MUD server in the manufacturing domain during the zero-touch onboarding process.
4. **Endorsement Keys**: an asymmetric (RSA) key-pair used to endorse claims that data is trustworthy (see Chapter 8 regarding trustworthiness). These keys are created when the TPM is manufactured and are not user specific. The key-pair is unique and remains fixed.
5. **DAA Keys**: privacy CA (DAA Issuer) asymmetric key-pair and signature key-pair created during secure enrolment. The protocol also creates attribute keys held by the issuer of the attributes and issued to the user (Prover), as well as authentication keys.
6. **Zero-Knowledge Keys**: keys used for interactions with the BC, to be leveraged in the DAA protocol of REWIRE.
7. **Data Sealing Keys**: sealed data is encrypted and authenticated in the Keystone TEE enclave, using SM and enclave related cryptographic keys, to enable secure and persistent storage of sensitive data. This process is necessary in the architecture of REWIRE, please see the proceeding subsection for detail.
8. **Migration Keys**: cryptographic keys used for the process of migrating the state of an enclave to another enclave, be that with the same host or a different one. This is necessary in REWIRE to maintain resilience within the heterogeneous environment it is designed for.
9. **Session Keys**: a symmetric, secret key randomly generated to ensure the security of a communication session between a user and device, or between two devices.

8.2 Integration with Keystone

Crucially, the KMS needs to be integrated with *Keystone TEE* to run critical services, such as key derivation, activation/deactivation, destruction, verification, and re-keying. Note that a KMS integrated with a TEE enables key generation in enclaves, whilst some keys can be generated in the SM which will act as a RoT for each KMS. Observe that these processes will be run in isolation from SW running on the platform. Components and communication channels in REWIRE need to be secure regardless of the network location. The security monitor (SM) will arbitrate communication between the untrusted world and the device. Despite SM in Keystone guaranteeing confidentiality and integrity of contents inside the enclave, it has been shown that the SM remains vulnerable to several attacks. Specifically, physical, software, side-channel, and DoS attacks. We defer the reader to Section 8.3 for details regarding protection of Keystone's SM and enclave with respect to the attacks.

The Keystone key-hierarchy is discussed in detail in [REF-218]. Here, the *root* key is an asymmetric processor key-pair, and the asymmetric SM key-pair is derived from a SM measurement as well as the secret key component of the root key-pair. Thus, the SM key-pair is bound to the SM identity and processor root key. Additional processes relevant to the key hierarchy includes *data sealing* which essentially allows an enclave to derive a key for encryption to be able to save data in untrusted, non-volatile memory outside of the enclave [REF-218]. Following the structure of Keystone's key hierarchy design, this means that a derived key is *bound* to the processor's identity, the SM, and the enclave. Assuming the processor, SM, and enclave remain the same, this means that the same key can be derived at a later point in time to decrypt encrypted data that has been outsourced in an untrusted environment. A less obvious feature of Keystone that REWIRE seeks to address is the migration of an enclave's state to another enclave on the same/different host. This presents several security and design challenges that must be considered in REWIRE. We need to address the possibility to dynamically generate migration keys from one enclave to another. Ensuring that enclaves co-operate is another challenge that can be addressed by having the policy orchestrator component of REWIRE generate a policy that enforces enclaves to perform their actions. The process of sealing a key requires a KDF taking the secret key component of the SM, a hash of the enclave and the keys identity as inputs. The output is the seal key which itself is input alongside the SM secret key into an ECDSA to output a key signature. The key

signature and seal key are the components of a Keystone Seal-Key Structure [REF-218]. Note that if the SM or processor changes, then the seal-key will be different as it is bound to the identities of both components. A hash of the enclave is also included for the same reasons, to bind the seal-key to the enclave, which means the seal-key will have to change when keys are migrated to another enclave as in REWIRE. To derive multiple keys from a singular enclave, Keystone dictates that the enclave can choose the key identity input into the KDF.

Currently, a simple Keystone supported attestation scheme using ed25519 signatures on hashes of the SM and enclave content [REF-219][REF-220]. The root SM public-key is needed alongside a hash of enclave, plus a hash of the expected SM and a copy of the enclave report after launch. Note that Keystone needs additional support to guarantee security of the device root keys, which should only be accessible by the SM.

8.3 Secure device on-boarding and key management requirements

Recall, Section 4.4.7 details the process and sequence of zero-touch onboarding (ZTO) and secure enrolment of a device into the REWIRE network. Zero touch onboarding is vital for the creation of key restriction usage policies which define appropriate use of keys for access control lists to resources and components in the REWIRE network. The heterogeneity of the REWIRE ecosystem uses an approach in which a device is onboarded during an authentication protocol utilising a manufacturing usage description (MUD) profile in an extensible authentication protocol [REF-221], merged with DAA. This approach of DAA leveraging ZKPs is needed to supplement EAP-AAA, since the protocol currently does not provide any privacy assurances. The policies are extracted from the blockchain, via secure oracles, by a privacy CA interacting with the manufacturing domain of the device such that successful onboarding only occurs if a device has correctly attested to a verifier that it is at an *expected state*, without releasing the actual state. Here, attestation enables remote authentication of the devices TPM (HW module) to the MUD verifier, whilst preserving the privacy of the platform (device).

Attribute-based attestation is a vital cryptographic mechanism for the secure enrolment a device to access mixed-criticality services in the network during runtime. During secure enrolment, a privacy CA (DAA issuer) issues a discrete logarithm representation (credential) of multiple attributes, each of which is associated with its own signing key held in a key hierarchy protected in the TPM. This means that attributes can be associated to an authorisation policy to restrict its usage, though this is not a necessity. A user can attest ownership of selected attributes in a DAA protocol, such that the attestation agent (within the device facility layer) augments credentials into verifiable presentations and proves authenticity in zero-knowledge to the MUD verifier. Secure enrolment is also recorded on the blockchain.

Self-sovereign and blockchain based access control [REF-222] needs to be considered in REWIRE, such that the secure oracle component in the architecture acts as the authenticating data feed of device runtime attestation to execute smart contracts whilst supporting privacy-preservation, using zero-knowledge of the self-issued attributes. The challenge in REWIRE will be injecting credentials into a verifiable presentation (VP) such that the credential was not issued by the privacy CA, rather, it represents a set of attributes related to the runtime state of the device, issued by the device itself. Self-issued credentials will be formed from a decentralized digital identifier (DID) document, stored in a BC-wallet, and used for identity verification on the blockchain. They are used to create a verifiable link between the device's unique identity and the associated credential. That is, the credentials contained in the VP (corresponding to runtime attestation) will be digitally signed by the device identity private key, generating a tamper-proof signature of the credential's authenticity. The working idea is to use a matrix of attributes, based on which attribute-based signatures (ABS) are created from some operation/predicate over a (subset) of a row in the matrix.

Chapter 9
Definition of the REWIRE MVP

This chapter provides the tables that map the use case requirements from the defined user stories (see Chapter 6) to the technical requirements of the REWIRE framework (see Chapter 5). The outcome is that the REWIRE framework is necessary to secure the next generation SoS during their entire lifecycle, from the design phase to runtime.

9.1 Mapping of use case requirements to REWIRE technical requirements

Under the guiding principle “Never Trust, Always Verify”, the goal of the envisioned REWIRE architecture is to enable the long-term transformation of the emerging SoS into a distributed smart ecosystem with embedded trust while demonstrating the use of formal verification, trusted computing, identify management, vulnerability detection and analysis and Blockchain technologies for addressing requirements of several vertical industry sectors (e.g., smart cities, automotive and smart satellites). The tables follow the coding convention from Chapter 6 for user stories (first row), and from Chapter 5 for REWIRE technical requirements (first column).

Table 9.1: Use Case #1 – Mapping use case requirements to technical requirements

Requirement ID	ODINS.US.1	ODINS.US.2	ODINS.US.3	ODINS.US.4	ODINS.US.5
Functional Requirements					
FR.FR.1			x	x	
FR.FR.2					
FR.FR.3		x	x		
FR.FR.4			x	x	
FR.FR.5	x	x	x		
FR.FR.6		x			
FR.FR.7		x			
FR.FR.8	x			x	
Security Requirements					
FR.FR.9	x	x			x
FR.FR.10	x	x	x		x
FR.FR.11	x	x	x		x
FR.FR.12	x	x			x
FR.FR.13	x				x
FR.FR.14	x			x	x
FR.FR.15	x				x
Operational Assurance Requirements					

FR.FR.16				x	
FR.FR.17	x	x	x	x	
FR.FR.18	x	x	x		
FR.FR.19		x	x		
FR.FR.20	x				
FR.FR.21		x			
FR.FR.22			x		
Formal verification Requirements					
FR.FR.23	x	x	x	x	x
FR.FR.24	x	x	x	x	x
FR.FR.25	x	x	x	x	x
FR.FR.26	x	x	x	x	x
RoT Requirements					
FR.FR.27	x	x	x	x	x
FR.FR.28	x	x	x	x	x
FR.FR.29	x	x	x	x	x
FR.FR.30	x	x	x	x	x
FR.FR.31	x	x	x	x	x

Table 9.2: Use Case #2 – Mapping use case requirements to technical requirements

Requirement ID	KEN.US.1	KEN.US.2	KEN.US.3	KEN.US.4
Functional Requirements				
FR.FR.1		x	x	
FR.FR.2	x	x		
FR.FR.3	x			
FR.FR.4		x	x	
FR.FR.5	x	x		
FR.FR.6		x	x	x
FR.FR.7				x
FR.FR.8	x			
Security Requirements				
FR.FR.9				
FR.FR.10				x
FR.FR.11				
FR.FR.12		x		x
FR.FR.13	x			

FR.FR.14	x			
FR.FR.15	x		x	
Operational Assurance Requirements				
FR.FR.16				
FR.FR.17			x	x
FR.FR.18			x	x
FR.FR.19		x		
FR.FR.20	x		x	
FR.FR.21		x		x
FR.FR.22	x			
Formal verification Requirements				
FR.FR.23	x	x	x	x
FR.FR.24	x	x	x	x
FR.FR.25	x	x	x	x
FR.FR.26	x	x	x	x
RoT Requirements				
FR.FR.27	x	x	x	x
FR.FR.28	x	x	x	x
FR.FR.29	x	x	x	x
FR.FR.30	x	x	x	x
FR.FR.31	x		x	x

Table 9.3: Use Case #3 – Mapping use case requirements to technical requirements

Requirement ID	LSF.US.1	LSF.US.2	LSF.US.3	LSF.US.4
Functional Requirements				
FR.FR.1				
FR.FR.2	x	x		x
FR.FR.3		x	x	x
FR.FR.4			x	
FR.FR.5	x			
FR.FR.6		x	x	x
FR.FR.7		x		
FR.FR.8				x
Security Requirements				
FR.FR.9	x	x		

FR.FR.10	x	x		
FR.FR.11	x	x		
FR.FR.12		x		
FR.FR.13				
FR.FR.14			x	
FR.FR.15				
Operational Assurance Requirements				
FR.FR.16			x	
FR.FR.17	x	x		
FR.FR.18	x	x		
FR.FR.19		x	x	
FR.FR.20	x	x		x
FR.FR.21	x			
FR.FR.22		x		
Formal verification Requirements				
FR.FR.23	x	x	x	x
FR.FR.24	x	x	x	x
FR.FR.25	x	x	x	x
FR.FR.26	x	x	x	x
RoT Requirements				
FR.FR.27	x	x	x	x
FR.FR.28	x	x	x	x
FR.FR.29	x	x	x	x
FR.FR.30	x	x	x	x
FR.FR.31	x	x	x	x

9.2 Prioritisation of Requirements

The REWIRE MVP depicts the shared vision of the consortium. A necessary step towards defining the REWIRE MVP is the prioritisation of the identified requirements. Thus, the REWIRE MVP will be built based on the following requirements set shown in the following table. The involvement of the demonstrators is crucial for establishing REWIRE's vision and guiding it toward the actual demands of the three use cases. The table below provides the mapping among the "must have" and "should" requirements of the MVP and its applicability per use case.

Table 9.4: REWIRE MVP

Requirement ID	Evident in all use cases	Evident in use case 1	Evident in use case 2	Evident in use case 3	Covered by no. of user stories

Functional Requirements					
FR.FR.1		x (2)	x (2)		4
FR.FR.2			x (2)	x (3)	5
FR.FR.3	x	x (2)	x (1)	x (3)	6
FR.FR.4	x	x (2)	x (2)	x (1)	5
FR.FR.5	x	x (3)	x (2)	x (1)	6
FR.FR.6	x	x (1)	x (3)	x (3)	7
FR.FR.7	x	x (1)	x (1)	x (1)	3
FR.FR.8	x	x (2)	x (1)	x (1)	4
Security Requirements					
FR.FR.9		x (3)		x (2)	5
FR.FR.10	x	x (4)	x (1)	x (2)	7
FR.FR.11		x (4)		x (2)	6
FR.FR.12	x	x (3)	x (2)	x (1)	6
FR.FR.13		x (2)	x (1)		3
FR.FR.14	x	x (3)	x (1)	x (1)	5
FR.FR.15		x (2)	x (2)		4
Operational Assurance Requirements					
FR.FR.16		x (1)		x (1)	2
FR.FR.17	x	x (4)	x (2)	x (2)	8
FR.FR.18	x	x (3)	x (2)	x (2)	7
FR.FR.19	x	x (2)	x (1)	x (2)	5
FR.FR.20	x	x (1)	x (2)	x (3)	6
FR.FR.21	x	x (1)	x (2)	x (1)	4
FR.FR.22	x	x (1)	x (1)	x (1)	3
Formal verification Requirements					
FR.FR.23	x				13
FR.FR.24	x				13
FR.FR.25	x				13
FR.FR.26	x				13
RoT Requirements					
FR.FR.27	x				13
FR.FR.28	x				13
FR.FR.29	x				13
FR.FR.30	x				13
FR.FR.31	x				13

The REWIRE MVP, consists of a total of **24 "must-have" requirements** (including 4 requirements for the formal verification and 5 for the underlying RoT), and 7 "should-have" requirements. The "must-have" requirements are suggested as the initial core requirements of the platform to be delivered (although the consortium's goal is to be able to validate all identified requirements) and are deemed the most important properties used in our three use cases.

Chapter 10
Conclusions

The deliverable provided an initial exploration and description of the state of the art for each technology module, defined the mandatory and desirable requirements and the conceptual architecture with a detailed breakdown of the business logic of each one of the comprised components and building blocks. REWIRE conceptual architecture aims to satisfy the requirements that have been formulated during the requirements analysis phase. The aforementioned exploration stems out of the vision of the consortium for a holistic security management framework that can safeguard “Systems-of-Systems” during their entire lifecycle, staring from the design phase with formal verification and theorem proving methods to the runtime phase with continuous authentication and verification. It has also defined in more detail the three use cases of the project, going down into the level of user stories that will be used to validate the overall REWIRE framework.

As a consequent step, the use cases have been linked with the technical requirements (e.g., functional, security etc.) that should be at least covered by the framework to be developed, and that will be tested by at least one of the three demonstrators. This list, alongside with the detailed descriptions of the use cases and their user stories, will be used as input to the design of the demonstrator’s plans, and will act as reference material for any consequent design and development activity of the project.

List of Abbreviations

Abbreviation	Translation
AADL	Architecture Analysis and Design Language
ABAC	Attribute-based Access Control
ABS	Attribute-based Signature
AC	Anonymous Credentials
ACL	Access-Control List
ADAS	Advanced Driver Assistance System
AGREE	Assume Guarantee REasoning Environment
AI	Artificial Intelligence
ASIL	Automotive Safety Integrity Level
AUTOSAR	AUTomotive Open System ARchitecture
BC	Blockchain
BMC	Bounded Model Checking
BSV	Bluespec SystemVerilog
BTL	Build-Test-Learn
CA	Certificate Authority
CAN	Controller Area Network
CAPEC	Common Attack Pattern Enumeration and Classification
CAV	Connected Autonomous Vehicle
CFA	Control Flow Attestation
CFG	Control Flow Graph
CIV	Configuration Integrity Verification
CNN	Convolutional Neural Networks
CPS	Cyber-Physical Systems
CTMD	Collaborative Threat and Misbehaviour Detection
CVSS	Common Vulnerability Scoring System
DAA	Direct Anonymous Attestation
DAA-A	Direct Anonymous Attestation with attributes
DCM	Diagnostic Communication Manager
DID	Decentralised ID
DL	Deep Learning
DLT	Distributed Ledger Technology
DMA	Direct Memory Access
DNN	Deep Neural Networks
DSL	Domain Specific Language

DT	Decision Tree
EAP	Extensible Authentication Protocol
EBP	European Blockchain Partnership
EBSI	European Blockchain Services Infrastructure
ECDAA	Elliptic Curve based DAA
ECU	Electronic Control Units
EK	Endorsement Key
EPC	Enclave Page Cache
EPCM	Enclave Page Cache Map
EPID	Enhanced Privacy ID
FIDO	Fast IDentity Online
FL	Federated Learning
FUOTA	Firmware Update Over the Air
FV	Formal Verification
GAN	Generative Adversarial Networks
HABS	Hierarchical Attribute-based Signature
HPC	High-Performance Computer
HW	Hardware
IAM	Identity & Access Management
INVEST	Independent, Negotiable, Valuable, Estimable, Small, and Testable
IOMMU	Input/Output Memory Management Unit
IPSec	Internet Protocol Security
ISA	Instruction Set Architecture
ISE	Instruction Set Extensions
JWT	Jason Web Tokens
KMS	Key Management System
KNN	K-Nearest Neighbour
LEO	Low Earth Orbit
LKAS	Lane-Keeping Assistance System
LR	Logistic Regression
LR	Linear Regression
MAC	Message Authentication Code
MBE	Model Based Engineering
ML	Machine Learning
MMU	Memory Management Unit
MITM	Man-in-the-Middle
MUD	Manufacturer Usage Description
MVP	Minimum Viable Product
NN	Neural Networks
OBD	On-Board Diagnostics

OEM	Original Equipment Manufacturer
PMP	Physical Memory Protection
PO	Policy Orchestrator
PSK	Pre-Shared Key
PSL	Property Specification Language
PTPS	Public Transportation Priority Systems
PUF	Physical Unclonable Functions
RA	Remote Attestation
REE	Rich Execution Environment
RF	Random Forest
RIA	Research and Innovation Actions
RNN	Recurrent Neural Networks
RoT	Root of Trust
RTL	Register Transfer Level
SecOC	Secure Onboard Communication
SEU	Single Events Upset
SGX	Software Guard Extension
SM	Security Monitor
SMT	Satisfiability Modulo Theory
SotA	State-of-the-Art
SPOF	Single Point of Failure
SSI	Self-Sovereign Identity
SoC	System on Chip
SOC	Service Oriented Communication
SoS	Systems-of-Systems
SVA	System Verilog Assertions
SVM	Support Vector Machine
SW	Software
TC	Trusted Component
TCB	Trusted Computing Base
TEE	Trusted Execution Environment
TLS	Transport Layer Security
TPM	Trusted Platform Module
TRL	Technology Readiness Level
TRNG	True Random Number Generator
UDS	Unified Diagnostic Service
VC	Verifiable Credential
VLR	Verifier-Local Revocation
VP	Verifiable Presentation
V&V	Validation and Verification

W3C	World Wide Web Consortium
WP	Work Package
ZKP	Zero-Knowledge Proof
ZTA	Zero-Trust Architecture
ZTO	Zero Trust On-boarding

References

Reference	Name of document
[REF-01]	Feiler, Peter H., David P. Gluch, and John J. Hudak. *The architecture analysis & design language (AADL): An introduction*. Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst, 2006.
[REF-02]	Feiler, Peter. *The open source aadl tool environment (osate)*. Carnegie Mellon University Software Engineering Institute, 2019.
[REF-03]	Cofer, Darren, et al. "Compositional verification of architectural models." *NASA Formal Methods: 4th International Symposium, NFM 2012, Norfolk, VA, USA, April 3-5, 2012. Proceedings 4*. Springer Berlin Heidelberg, 2012.
[REF-04]	Gacek, Andrew, et al. "Resolute: an assurance case language for architecture models." *ACM SIGAda Ada Letters* 34.3 (2014): 19-28.
[REF-05]	Denney, Ewen, and Ganesh Pai. "Tool support for assurance case development." *Automated Software Engineering* 25.3 (2018): 435-499.
[REF-06]	Hause, Matthew. "The SysML modelling language." *Fifteenth European Systems Engineering Conference*. Vol. 9. 2006.
[REF-07]	Cofer, Darren, et al. "Cyberassured systems engineering at scale." *IEEE Security & Privacy* 20.3 (2022): 52-64.
[REF-08]	Klein, Gerwin, et al. "seL4: Formal verification of an OS kernel." *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. 2009.
[REF-09]	The Coq Proof Assistant. [online] Available at: https://coq.inria.fr/ [Accessed 11 Nov. 2023].
[REF-10]	Chlipala, Adam. *Certified programming with dependent types: a pragmatic introduction to the Coq proof assistant*. MIT Press, 2022.
[REF-11]	Leroy, Xavier, et al. "CompCert-a formally verified optimizing compiler." *ERTS 2016: Embedded Real Time Software and Systems, 8th European Congress*. 2016.
[REF-12]	Choi, Joonwon, et al. "Kami: a platform for high-level parametric hardware specification and its modular verification." (2017).
[REF-13]	Bourgeat, Thomas, Clément Pit-Claudel, and Adam Chlipala. "The essence of Bluespec: a core language for rule-based hardware design." *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation*. 2020.
[REF-14]	Nikhil, Rishiyur. "Bluespec System Verilog: efficient, correct RTL from high level specifications." *Proceedings. Second ACM and IEEE International Conference on Formal Methods and Models for Co-Design, 2004. MEMOCODE'04.*. IEEE, 2004.
[REF-15]	Pnueli, Amir. "The temporal logic of programs." *In 18th Annual Symposium on Foundations of Computer Science (sfcs 1977), pages 46–57*. ieee, 1977.
[REF-16]	Clarke, Edmund et al. "Design and synthesis of synchronization skeletons using branching time temporal logic." 1981.
[REF-17]	Queille, Jean-Pierre et al. "Specification and verification of concurrent systems in cesar." *In International Symposium on Programming: 5th Colloquium Turin, April 6–8, 1982 Proceedings, pages 337–351*. Springer, 2005.
[REF-18]	Burch, Jerry et al. Symbolic model checking: 1020 states and beyond. *Information and computation, 98(2):142–170*, 1992.
[REF-19]	Bryant, Randal. "Graph-based algorithms for boolean function manipulation." *Computers, IEEE Transactions on, 100(8):677–691*, 1986.
[REF-20]	Marques-Silva, Joao et al. "Conflict-driven clause-learning sat solvers." *In Handbook of satisfiability, pages 133–182*. IOS press, 2021.
[REF-21]	Biere, Armin et al. "Symbolic model checking without bdds." *In Tools and Algorithms for the Construction and Analysis of Systems: 5th International Conference, TACAS'99 Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS'99 Amsterdam, The Netherlands, March 22–28, 1999 Proceedings 5, pages 193–207.* Springer, 1999.
[REF-22]	Sheeran, Mary et al. "Checking safety properties using induction and a sat-solver." *In Formal Methods in Computer-Aided Design: Third International Conference, FMCAD 2000 Austin, TX, USA, November 1–3, 2000 Proceedings 3, pages 127–144.* Springer, 2000.
[REF-23]	McMillan, Kenneth. "Interpolation and sat-based model checking." *In Computer Aided Verification: 15th International Conference, CAV 2003, Boulder, CO, USA, July 8-12, 2003. Proceedings 15, pages 1–13.* Springer, 2003.
[REF-24]	Clarke, Edmund et al. "Counterexample-guided abstraction refinement for symbolic model checking." Journal of the ACM (JACM), 50(5):752–794, 2003.

[REF-25]	Bradley, Aaron. “Sat-based model checking without unrolling.” *In Verification, Model Checking, and Abstract Interpretation: 12th International Conference, VMCAI 2011, Austin, TX, USA, January 23-25, 2011. Proceedings 12, pages 70–87.* Springer, 2011.
[REF-26]	Een, Niklas et al. “Efficient implementation of property directed reachability.” *In 2011 Formal Methods in Computer-Aided Design (FMCAD), pages 125–134.* IEEE, 2011.
[REF-27]	Preiner, Mathias et al. “Hardware model checking competition 2020.” 2020.
[REF-28]	IEEE-Commission et al. “Ieee standard for property specification language (psl).” *IEEE Std 1850-2005,* 2005.
[REF-29]	Design Automation Standards Committee et al. “Ieee standard for systemverilog unified hardware design, specification, and verification language standard ieee 1800.” *http://www. edastds. org/sv/*, 2005.
[REF-30]	Rajendran, Jeyavijayan et al. “Detecting malicious modifications of data in third-party intellectual property cores.” *In 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), pages 1–6*, 2015.
[REF-31]	Cruz, Jonathan et al. “Hardware trojan detection using atpg and model checking.” *In 2018 31st International Conference on VLSI Design and 2018 17th International Conference on Embedded Systems (VLSID), pages 91–96*, 2018.
[REF-32]	Guo, Xiaolong et al. “Scalable soc trust verification using integrated theorem proving and model checking.” *In 2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pages 124–129*, 2016.
[REF-33]	Rathmair, Michael et al. “Applied formal methods for hardware trojan detection.” *In 2014 IEEE International Symposium on Circuits and Systems (ISCAS), pages 169–172,* 2014.
[REF-34]	Hu, Wei et al. “Detecting hardware trojans with gate-level information-flow tracking.” *Computer, 49(8):44–52*, 2016.
[REF-35]	Seshia, Sanjit et al. “Uclid5: Integrating modeling, verification, synthesis, and learning.” *In Proceedings of the 15th ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMOCODE),* October 2018.
[REF-36]	Chipyard Framework. [online] Available at: https://github.com/ucb-bar/chipyard [Accessed 11 Nov. 2023].
[REF-37]	Asanović, Krste et al “ The Rocket Chip Generator”, EECS Department, University of California, Berkeley, Technical report 2016, UCB/EECS-2016-17
[REF-38]	RISC-V. [online] Available at: https://riscv.org/ [Accessed 11 Nov. 2023].
[REF-39]	The Sail ISA specification language. [online] Available at: https://github.com/rems-project/sail [Accessed 11 Nov. 2023].
[REF-40]	Black, J. (2004). Authenticated Encryption. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=2628d946bda9f3d3b087e5c4846e76ae0fb07b6b
[REF-41]	Crypto competitions: CAESAR call for submissions, final (2014.01.27). (2014). Cr.yp.to. https://competitions.cr.yp.to/caesar-call.html
[REF-42]	Standaert, F.-X. (n.d.). Introduction to Side-Channel Attacks. Retrieved July 25, 2023, from https://perso.uclouvain.be/fstandae/PUBLIS/42.pdf
[REF-43]	Kocher, P., Jaffe, J., & Jun, B. (1999). Differential Power Analysis. 388–397. https://doi.org/10.1007/3-540-48405-1_25
[REF-44]	Smart Card Security: Countering Side Channel and Fault Attacks | NXP Semiconductors. (2020). @NXP. https://www.nxp.com/design/training/smart-card-security-countering-side-channel-and-fault-attacks:TIP-SMART-CARD-SECURITY
[REF-45]	Lightweight Cryptography | CSRC. Nist.gov. https://csrc.nist.gov/Projects/lightweight-cryptography
[REF-46]	Dobraunig, C., Eichlseder, M., Mendel, F., & Schläffer, M. (2019). Submission to NIST. https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/spec-doc-rnd2/ascon-spec-round2.pdf
[REF-47]	Bellizia, D., Bronchain, O., Cassiers, G., Grosso, V., Guo, C., Momin, C., Pereira, O., Peters, T., & Standaert, F.-X. (n.d.). Mode-Level vs. Implementation-Level Physical Security in Symmetric Cryptography A Practical Guide Through the Leakage-Resistance Jungle. Retrieved July 25, 2023, from https://eprint.iacr.org/2020/211.pdf
[REF-48]	Pereira, O., Peters, T., & Standaert, F.-X. (2023). Mode-Level Side-Channel Countermeasures. https://perso.uclouvain.be/fstandae/PUBLIS/296.pdf

[REF-49]	Tomoaki Ukezono. (2021). Resistance for Side-Channel Attack by Virtual Dual-Rail Effect. https://doi.org/10.1109/icecce52056.2021.9514176
[REF-50]	Gross, H., Mangard, S., & Korak, T. (n.d.). Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order. https://eprint.iacr.org/2016/486.pdf
[REF-51]	Veyrat-Charvillon, N., Medwed, M., Stéphanie Kerckhof, & Standaert, F.-X. (2012). Shuffling against Side-Channel Attacks: A Comprehensive Study with Cautionary Note. 740–757. https://doi.org/10.1007/978-3-642-34961-4_44
[REF-52]	IsapSpecification. [online] Available at: https://isap.iaik.tugraz.at/specification.html [Accessed 11 Nov. 2023].
[REF-53]	Improved Leakage-Resistant Authenticated Encryption based on Hardware AES Coprocessors. [online] Available at: https://tches.iacr.org/index.php/TCHES/article/view/8988 [Accessed 11 Nov. 2023].
[REF-54]	Intel 64 and IA-32 Architectures Optimization Reference Manual, Intel, [online] Available at: https://software.intel.com/content/www/us/en/develop/download/intel-64-and-ia-32-architectures-optimization-reference-manual.html, 2018 [Accessed 11 Nov. 2023].
[REF-55]	Intel Software Guard Extensions Developer Guide, Intel, https://software.intel.com/content/www/us/en/develop/download/intel-software-guard-extensions-intel-sgx-developer-guide.html?wapkw=SGX, 2018
[REF-56]	Intel SGX Explained, V. Costan and S. Devadas, IACR Cryptol. ePrint Arch., 2016/086, 2016
[REF-57]	Intel Software Guard Extensions Remote Attestation, Intel, [online] Available at: https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/attestation-services.html, 2018 [Accessed 11 Nov. 2023].
[REF-58]	Enhanced Privacy ID: A Direct Anonymous Attestation Scheme with Enhanced Revocation Capabilities, E. Brickell, J. Li, IEEE Trans. Dependable Secur. Comput. 9(3): 345-360, 2012
[REF-59]	ROTE: Rollback Protection for Trusted Execution, S. Matetic, M. Ahmed, K. Kostiainen, A. Dhar, D. Sommer, A. Gervais, A. Juels, S. Capkun , USENIX Security Symposium, pp 1289-1306, 2017
[REF-60]	Software Grand Exposure: SGX Cache Attacks Are Practical, F. Brasser, U. Müller, A. Dmitrienko, K. Kostiainen, S. Capkun, A.R. Sadeghi, USENIX Workshop on Offensive Technologies, 2017
[REF-61]	Preventing Page Faults from Telling Your Secrets, S. Shinde, Z. Leong Chua, V. Narayanan, P. Saxena, AsiaCCS, pp 317-328, 2016
[REF-62]	SgxPectre: Stealing Intel Secrets from SGX Enclaves Via Speculative Execution, G. Chen, S. Chen, Y. Xiao, Y. Zhang, Z. Lin, T-H. Lai, EuroS&P, pp 142-157, 2019
[REF-63]	The RISC-V Instruction Set Manual, Volume 2, Privileged Specification version 20211203. Available: https://github.com/riscv/riscv-isa-manual/releases/download/Priv-v1.12/riscv-privileged-20211203.pdf
[REF-64]	Sanctum: Minimal hardware extensions for strong software isolation. V. Costan, I. Lebedev, and S. Devadas. In USENIX Security, 2016.
[REF-65]	Keystone: an open framework for architecting trusted execution environments. Dayeol Lee, David Kohlbrenner, Shweta Shinde, Krste Asanović, and Dawn Song. 2020. In Proceedings of the Fifteenth European Conference on Computer Systems (EuroSys '20). Association for Computing Machinery, New York, NY, USA, Article 38, 1–16. https://doi.org/10.1145/3342195.3387532
[REF-66]	CURE: A Security Architecture with CUstomizable and Resilient Enclaves. Raad Bahmani, Ferdinand Brasser, Ghada Dessouky, Patrick Jauernig, Matthias Klimmek, Ahmad-Reza Sadeghi, Emmanuel Stapf. CoRR abs/2010.15866 (2020)
[REF-67]	Komodo: Using verification to disentangle secure-enclave hardware from software. A. Ferraiuolo, A. Baumann, C. Hawblitzel, and B. Parno. In SOSP, pages 287–305. ACM, 2017.
[REF-68]	GlobalPlatform. [online] Available at: https://globalplatform.org/specs-library/?filter-committee=tee [Accessed 11 Nov. 2023].
[REF-69]	Open-TEE project. https://github.com/Open-TEE/project
[REF-70]	Dayeol Lee, David Kohlbrenner, Shweta Shinde, Krste Asanović, and Dawn Song. 2020. Keystone: An Open Framework for Architecting Trusted Execution Environments. In Fifteenth European Conference on Computer Systems (EuroSys '20), April 27–30, 2020, Heraklion, Greece. ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3342195.3387532

[REF-71]	How-Keystone-Works/RISC-V-Background.rst RISC-V Privileged ISA https://github.com/keystone-enclave/keystone/blob/master/docs/source/Getting-Started/How-Keystone-Works/RISC-V-Background.rst
[REF-72]	Open Portable TEE. 2020. https://www.op-tee.org/
[REF-73]	G. Coker, J. Guttman, P. Loscocco, A. Herzog, J. Millen, B. O'Hanlon, H. Ramsdell, A. Segall, J. Sheehy, and B. Sniffen, "Principles of remote attestation," International Journal of Information Security, vol. 10, pp. 63–81, 2011.
[REF-74]	A. Segall, Trusted Platform Modules: Why, when and how to use them. London, United Kingdom: Institution of Engineering and Technology, 2017.
[REF-75]	K. Kostiainen, A. Dmitrienko, J.-E. Ekberg, A.-R. Sadeghi, and N. Asokan, "Key attestation from trusted execution environments," in Trust and Trustworthy Computing, ser. Lecture Notes in Computer Science, vol 6101. Berlin, Heidelberg: Springer, 2010, pp. 30–46.
[REF-76]	Seema Kumar, Patrick Eugster, and Silvia Santini. 2021. Software-based Remote Network Attestation. IEEE Transactions on Dependable and Secure Computing 19, 5 (2021), 1–1. https://doi.org/10.1109/TDSC.2021.3077993
[REF-77]	Ernie Brickell, Jan Camenisch, and Liqun Chen. 2004. Direct Anonymous Attestation. In Proceedings of the 11th ACM Conference on Computer and Communications Security (Washington DC, USA) (CCS '04). Association for Computing Machinery, New York, NY, USA, 132–145. https://doi.org/10.1145/1030083.1030103
[REF-78]	Guoxing Chen and Yinqian Zhang. 2022. MAGE: Mutual Attestation for a Group of Enclaves without Trusted Third Parties. In 31st USENIX Security Symposium (USENIX Security 22). USENIX Association, Boston, MA, 4095–4110.
[REF-79]	B. Larsen, H. B. Debes, and T. Giannetsos, "Cloudvaults: Integrating trust extensions into system integrity verification for cloud-based environments," in European Symposium on Research in Computer Security. Springer, 2020, pp. 197–220.
[REF-80]	M. Abadi, M. Budiu, U. Erlingsson and J. Ligatti, "Control-flow Integrity Principles Implementations and Applications", Proceedings of ACM Transactions on Information and System Security, vol. 13, no. 1, pp. 1-40, 2009.
[REF-81]	K. Z. Snow, F. Monrose, L. Davi, A. Dmitrienko, C. Liebchen and A.-R. Sadeghi, "Just-in-time Code Reuse: On the Effectiveness of Finegrained Address Space Layout Randomization", *Proceedings of IEEE Symposium on Security and Privacy*, pp. 574-588, 2013.
[REF-82]	R. Roemer, E. Buchanan, H. Shacham and S. Savage, "Return-oriented Programming: Systems Languages and Applications", Proceedings of ACM Transactions on Information and System Security, vol. 15, no. 1, pp. 1-34, 2012.
[REF-83]	T. Abera, N. Asokan, L. Davi, J.-E. Ekberg, T. Nyman, A. Paverd, et al., "C-FLAT: Control-Flow Attestation for Embedded Systems Software", *Proceedings of ACM SIGSAC Conference on Computer and Communications Security*, pp. 743-754, 2016.
[REF-84]	Y. Zhang, X. Liu, C. Sun, D. Zeng, G. Tan, X. Kan, et al., "ReCFA: Resilient Control-Flow Attestation", Proceedings of Computer Security Applications Conference, pp. 311-322, 2021.
[REF-85]	ftrace - Function Tracer, https://www.kernel.org/doc/html/v5.0/trace/ftrace.html
[REF-86]	Nikos Koutroumpouchos, Christoforos Ntantogian, Sofia-Anna Menesidou, Kaitai Liang, Panagiotis Gouvas, Christos Xenakis, and Thanassis Giannetsos. Secure edge computing with lightweight control-flow property-based attestation. In 2019 IEEE Conference on Network Softwarization (NetSoft), pages 84–92, 2019
[REF-87]	Oracle Solaris 11.4 DTrace (Dynamic Tracing) Guide, https://docs.oracle.com/en/operating-systems/solaris/oracle-solaris/11.4/dtrace-guide/oracle-solaris-11.4-dtrace-dynamic-tracing-guide.pdf
[REF-88]	LTTng. [online] Available at: https://lttng.org/ [Accessed 11 Nov. 2023].
[REF-89]	Suchakrapani Datt Sharma and Michel Dagenais. Enhanced userspace and in-kernel trace filtering for production systems. Journal of Computer Science and Technology, 31(6):1161– 1178, 2016.
[REF-90]	M. Davie, D. Gisolfi, D. Hardman, J. Jordan, D. O'Donnell and D. Reed, "The Trust over IP Stack," in *IEEE Communications Standards Magazine*, vol. 3, no. 4, pp. 46-51, December 2019, doi: 10.1109/MCOMSTD.001.1900029.
[REF-91]	Decentralized Identifiers (DIDs) v1.0. [online] Available at: https://www.w3.org/TR/did-core/ [Accessed 11 Nov. 2023].
[REF-92]	DIDComm Messaging v2.x Editor's Draft. [online] Available at: https://identity.foundation/didcomm-messaging/spec/ [Accessed 11 Nov. 2023].
[REF-93]	A. Fraser and S. Schneider. On the role of blockchain for self-sovereign identity. In Competitive Advantage in the Digital Economy (CADE 2022), volume 2022, pages 17–21, 2022.
[REF-94]	https://www.w3.org/TR/vc-data-model-2.0/

[REF-95]	D. W. Chadwick, R. Laborde, A. Oglaza, R. Venant, S. Wazan and M. Nijjar, "Improved Identity Management with Verifiable Credentials and FIDO," in *IEEE Communications Standards Magazine*, vol. 3, no. 4, pp. 14-20, December 2019, doi: 10.1109/MCOMSTD.001.1900020.
[REF-96]	DID Specification Registries. [online] Available at: https://www.w3.org/TR/did-spec-registries/#verification-method-types [Accessed 11 Nov. 2023].
[REF-97]	Linked Data Cryptographic Suite Registry. [online] Available at: https://w3c-ccg.github.io/ld-cryptosuite-registry/ [Accessed 11 Nov. 2023].
[REF-98]	Experience cross-borders services with EBSI. [online] Available at: https://ec.europa.eu/digital-building-blocks/wikis/display/EBSI/Home [Accessed 11 Nov. 2023].
[REF-99]	Introduction to Microsoft Entra Verified ID. [online] Available at: https://learn.microsoft.com/en-us/azure/active-directory/verifiable-credentials/decentralized-identifier-overview [Accessed 11 Nov. 2023].
[REF-100]	AnonCreds Specification. [online] Available at: https://hyperledger.github.io/anoncreds-spec/ [Accessed 11 Nov. 2023].
[REF-101]	Kakvi, Saqib A., et al. "SoK: Anonymous Credentials." *International Conference on Research in Security Standardisation*. Cham: Springer Nature Switzerland, 2023.
[REF-102]	B. Podgorelec, L. Alber and T. Zefferer, "What is a (Digital) Identity Wallet? A Systematic Literature Review," 2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC), Los Alamitos, CA, USA, 2022, pp. 809-818, doi: 10.1109/COMPSAC54236.2022.00131.
[REF-103]	Ansaroudi, Z.E., Carbone, R., Sciarretta, G., Ranise, S. (2023). Control is Nothing Without Trust a First Look into Digital Identity Wallet Trends. In: Atluri, V., Ferrara, A.L. (eds) Data and Applications Security and Privacy XXXVII. DBSec 2023. Lecture Notes in Computer Science, vol 13942. Springer, Cham. https://doi.org/10.1007/978-3-031-37586-6_7
[REF-104]	Steffen Schwalm, and Ignacio Alamillo-Domingo, Journal: Wirtschaftsinformatik 2021, “Self-Sovereign-Identity \& eIDAS: a Contradiction? Challenges and Chances of eIDAS 2.0”
[REF-105]	Voskobojnikov, Artemij, et al. "The u in crypto stands for usable: An empirical study of user experience with mobile cryptocurrency wallets." *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 2021.
[REF-106]	Barakat, Samer, Qais Hammouri, and Khalil Yaghi. "COMPARISON OF HARDWARE AND DIGITAL CRYPTO WALLETS." *Journal of Southwest Jiaotong University* 57.6 (2022).
[REF-107]	S. R. Tate and R. Vishwanathan, "Performance evaluation of TPM-based digital wallets," *Proceedings of the 2010 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS '10)*, Ottawa, ON, Canada, 2010, pp. 179-186.
[REF-108]	Johnson Jeyakumar, Isaac H.; Chadwick, David W.; Kubach, Michael (2022): A novel approach to establish trust in verifiable credential issuers in Self-sovereign identity ecosystems using TRAIN. Open Identity Summit 2022. DOI: 10.18420/OID2022_02. Bonn: Gesellschaft für Informatik e.V.. PISSN: 1617-5468. ISBN: 978-3-88579-719-7. pp. 27-38. Regular Research Papers. Copenhagen, Denmark. 07.-08. July 2022
[REF-109]	Binwalk. [online] Available at: https://github.com/ReFirmLabs/binwalk [Accessed 11 Nov. 2023].
[REF-110]	Binary Analysis Next Generation (BANG). [online] Available at: https://github.com/armijnhemel/binaryanalysis-ng [Accessed 11 Nov. 2023].
[REF-111]	American Fuzzy Lop plus plus (AFL++). [online] Available at: https://github.com/AFLplusplus/AFLplusplus [Accessed 11 Nov. 2023].
[REF-112]	Qiling's usecase, blog and related work. [online] Available at: https://github.com/qilingframework/qiling [Accessed 11 Nov. 2023].
[REF-113]	Unicorn The Ultimate CPU emulator. [online] Available at: https://www.unicorn-engine.org/ [Accessed 11 Nov. 2023].
[REF-114]	Capstone The Ultimate Disassembler. [online] Available at: http://www.capstone-engine.org/showcase.html [Accessed 11 Nov. 2023].
[REF-115]	Keystone The Ultimate Assembler. [online] Available at: https://www.keystone-engine.org/ [Accessed 11 Nov. 2023].
[REF-116]	angr. [online] Available at: https://angr.io/ [Accessed 11 Nov. 2023].
[REF-117]	Abadi, M., Budiu, M., Erlingsson, U., & Ligatti, J. (2009). Control-flow integrity principles, implementations, and applications. *ACM Transactions on Information and System Security (TISSEC)*, *13*(1), 1-40.
[REF-118]	RISC-V CFI specification. [online] Available at: https://github.com/riscv/riscv-cfi [Accessed 11 Nov. 2023].
[REF-119]	Gilles, O., Viguier, F., Kosmatov, N., & Pérez, D. G. (2022). Control-flow integrity at risc: Attacking risc-v by jump-oriented programming. *arXiv preprint arXiv:2211.16212*.

[REF-120]	Zhang, M., Qiao, R., Hasabnis, N., & Sekar, R. (2014, March). A platform for secure static binary instrumentation. In *Proceedings of the 10th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments* (pp. 129-140).
[REF-121]	Salehi, M., Hughes, D., & Crispo, B. (2020). {μSBS}: Static Binary Sanitization of Bare-metal Embedded Devices for Fault Observability. In *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)* (pp. 381-395).
[REF-122]	Chen, C., Jing, Z., Ma, J., Cui, B., Xu, H., & Zou, Q. (2020). IoTSIT: A Static Instrumentation Tool for IoT Devices. *IEEE Access*, *8*, 92153-92161.
[REF-123]	Chen, L., Yuan, R. and Xia, Y., 2021, August. Tora: A trusted blockchain oracle based on a decentralized tee network. In *2021 IEEE International Conference on Joint Cloud Computing (JCC)* (pp. 28-33). IEEE.
[REF-124]	Basile, D., Goretti, V., Di Ciccio, C. and Kirrane, S., 2021, August. Enhancing blockchain-based processes with decentralized oracles. In *Business Process Management: Blockchain and Robotic Process Automation Forum: BPM 2021 Blockchain and RPA Forum, Rome, Italy, September 6–10, 2021, Proceedings* (pp. 102-118). Cham: Springer International Publishing.
[REF-125]	Eskandari, S., Salehi, M., Gu, W.C. and Clark, J., 2021, September. SoK: oracles from the ground truth to market manipulation. In *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies* (pp. 127-141).
[REF-126]	Pasdar, A., Lee, Y.C. and Dong, Z., 2023. Connect api with blockchain: A survey on blockchain oracle implementation. ACM Computing Surveys, 55(10), pp.1-39.
[REF-127]	Singh, A., Click, K., Parizi, R.M., Zhang, Q., Dehghantanha, A. and Choo, K.K.R., 2020. Sidechain technologies in blockchain networks: An examination and state-of-the-art review. Journal of Network and Computer Applications, 149, p.102471.
[REF-128]	Luu, L., Narayanan, V., Zheng, C., Baweja, K., Gilbert, S. and Saxena, P., 2016, October. A secure sharding protocol for open blockchains. In Proceedings of the 2016 ACM SIGSAC conference on computer and communications security (pp. 17-30).
[REF-129]	Negka, L.D. and Spathoulas, G.P., 2021. Blockchain state channels: A state of the art. IEEE Access, 9, pp.160277-160298.
[REF-130]	Macrinici, D., Cartofeanu, C. and Gao, S., 2018. Smart contract applications within blockchain technology: A systematic mapping study. Telematics and Informatics, 35(8), pp.2337-2354.
[REF-131]	Foschini, L., Gavagna, A., Martuscelli, G. and Montanari, R., 2020, June. Hyperledger fabric blockchain: Chaincode performance analysis. In ICC 2020-2020 IEEE International Conference on Communications (ICC) (pp. 1-6). IEEE.
[REF-132]	Shi, Z., Zhou, H., Hu, Y., Jayachander, S., de Laat, C. and Zhao, Z., 2019, June. Operating permissioned blockchain in clouds: A performance study of hyperledger sawtooth. In 2019 18th International Symposium on Parallel and Distributed Computing (ISPDC) (pp. 50-57). IEEE.
[REF-133]	TEEvault Blockchain Key Management Solution. [online] Available at: https://teeware.io/product/ [Accessed 11 Nov. 2023].
[REF-134]	Lux, Z.A., Thatmann, D., Zickau, S. and Beierle, F., 2020, September. Distributed-ledger-based authentication with decentralized identifiers and verifiable credentials. In 2020 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS) (pp. 71-78). IEEE.
[REF-135]	Rouhani, S. and Deters, R., 2019, October. Blockchain based access control systems: State of the art and challenges. In IEEE/WIC/ACM International Conference on Web Intelligence (pp. 423-428).
[REF-136]	Surjandari, I., Yusuf, H., Laoh, E. and Maulida, R., 2021. Designing a Permissioned Blockchain Network for the Halal Industry using Hyperledger Fabric with multiple channels and the raft consensus mechanism. Journal of Big Data, 8(1), pp.1-16.
[REF-137]	Spectra. [online] Available at: https://spectralogic.com/features/data-integrity-verification/ [Accessed 11 Nov. 2023].
[REF-138]	Bergers, J., Shi, Z., Korsmit, K. and Zhao, Z., 2021, December. Dwh-dim: a blockchain based decentralized integrity verification model for data warehouses. In 2021 IEEE International Conference on Blockchain (Blockchain) (pp. 221-228). IEEE.
[REF-139]	BOUALOUACHE, Abdelwahab; ENGEL, Thomas. A survey on machine learning-based misbehaviour detection systems for 5g and beyond vehicular networks. IEEE Communications Surveys & Tutorials, 2023.
[REF-140]	VAN DER HEIJDEN, Rens Wouter, et al. Survey on misbehaviour detection in cooperative intelligent transportation systems. _IEEE Communications Surveys & Tutorials, 2018, 21.1: 779-811.

[REF-141]	XU, Xiaoya; WANG, Yunpeng; WANG, Pengcheng. Comprehensive review on misbehaviour detection for vehicular ad hoc networks. Journal of Advanced Transportation, 2022, 2022.
[REF-142]	GHANBARI, Zahra, et al. Resource allocation mechanisms and approaches on the Internet of Things. Cluster Computing, 2019, 22.4: 1253-1282.
[REF-143]	MBAREK, Bacem, et al. A secure authentication mechanism for resource constrained devices. In: 2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA). IEEE, 2015. p. 1-7.
[REF-144]	LYU, Lingjuan, et al. Threats to federated learning. Federated Learning: Privacy and Incentive, 2020, 3-16.
[REF-145]	FOTIADOU, Konstantina, et al. Network traffic anomaly detection via deep learning. Information, 2021, 12.5: 215.
[REF-146]	GUPTA, Rajesh, et al. Machine learning models for secure data analytics: A taxonomy and threat model. Computer Communications, 2020, 153: 406-440.
[REF-147]	AYTEKIN, Caglar, et al. Clustering and unsupervised anomaly detection with l 2 normalized deep auto-encoder representations. In: 2018 International Joint Conference on Neural Networks (IJCNN). IEEE, 2018. p. 1-6.
[REF-148]	PAPALEXAKIS, Evangelos E.; BEUTEL, Alex; STEENKISTE, Peter. Network anomaly detection using co-clustering. In: _2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining_. IEEE, 2012. p. 403-410.
[REF-149]	ERGEN, Tolga; KOZAT, Suleyman Serdar. Unsupervised anomaly detection with LSTM neural networks. IEEE transactions on neural networks and learning systems, 2019, 31.8: 3127-3141.
[REF-150]	GE, Mengmeng, et al. Deep learning-based intrusion detection for IoT networks. In: 2019 IEEE 24th pacific rim international symposium on dependable computing (PRDC). IEEE, 2019. p. 256-25609.
[REF-151]	Hajiheidari, S., Wakil, K., Badri, M. and Navimipour, N.J., 2019. Intrusion detection systems in the Internet of things: A comprehensive investigation. Computer Networks, 160, pp.165-191.
[REF-152]	K. Beck, M. Beedle, A. Van Bennekum, A. Cockburn, W. Cunningham, M. Fowler and D. Thomas, “The agile manifesto.”, 2001
[REF-153]	D. S. U. Pandey and A. K. Ramani, “An effective requirement engineering process model for software development and requirements management”, in International Conference on Advances in Recent Technologies in Communication and Computing, 2010
[REF-154]	M. Cohn, User stories applied: for agile software development, 2004
[REF-155]	[2] Matheu Garcia, S., Molina Zarca, A., Hernandez Ramos, J., Bernal Bernabe, J. and Skarmeta, A., Enforcing behavioural profiles through Software-Defined Networks in the Industrial Internet of Things, APPLIED SCIENCES-BASEL, ISSN 2076-3417, 9 (21), 2019, p. 4576, JRC117341. https://publications.jrc.ec.europa.eu/repository/handle/111111111/58255
[REF-156]	charter-ietf-suit-02. IETF Software Updates for Internet of Things WG https://datatracker.ietf.org/wg/suit/about/
[REF-157]	Göran Selander, John Preuß Mattsson, Francesca Palombini, and Ludwig Seitz. Object Security for Constrained RESTful Environments (OSCORE). RFC 8613, July 2019. URL https://www.rfc-editor.org/info/rfc8613
[REF-158]	Marco Tiloca, Göran Selander, Francesca Palombini, John Preuß Mattsson, and Jiye Park. Group Object Security for Constrained RESTful Environments (Group OSCORE). Internet-Draft draft-ietf-core-oscore-groupcomm-19, Internet Engi- neering Task Force, July 2023. URL https://datatracker.ietf.org/doc/draft-ietf-core-oscore-groupcomm/19/. Work in Progress.
[REF-159]	Göran Selander, John Preuß Mattsson, Mališa Vučinić, Michael Richardson, and Aurelio Schellenbaum. Lightweight Authorization for EDHOC. Internet-Draft draft-selander-lake-authz-00, Internet Engineering Task Force, October 2022. URL https://datatracker.ietf.org/doc/draft-selander-lake-authz/00/. Work in Progress.
[REF-160]	Scott W. Rose, Oliver Borchert, Stuart Mitchell, Sean Connelly - Zero Trust Architecture, NIST Tech Report 2020, SP 800-207 (DOI)
[REF-161]	A. Fraser and S. Schneider. On the role of blockchain for self-sovereign identity. In Competitive Advantage in the Digital Economy (CADE 2022), volume 2022, pages 17–21, 2022.
[REF-162]	Kakvi, S.A., Martin, K.M., Putman, C., Quaglia, E.A. (2023). SoK: Anonymous Credentials. In: Günther, F., Hesse, J. (eds) Security Standardisation Research. SSR 2023. Lecture Notes in Computer Science, vol 13895. Springer, Cham. https://doi.org/10.1007/978-3-031-30731-7_6
[REF-163]	Hierarchical attribute-based signatures: Constantin-Cătălin Drăgan, and Daniel Gardham, and Mark Manulis - Cryptology and Network Security (CANS 2018)

[REF-164]	Zero-Knowledge Proofs. [online] Available at: https://www.w3.org/TR/vc-data-model/#zero-knowledge-proofs [Accessed 11 Nov. 2023].
[REF-165]	Gardham, D., Manulis, M. (2022). Revocable Hierarchical Attribute-Based Signatures from Lattices. In: Ateniese, G., Venturi, D. (eds) Applied Cryptography and Network Security. ACNS 2022. Lecture Notes in Computer Science, vol 13269. Springer, Cham. https://doi.org/10.1007/978-3-031-09234-3_23
[REF-166]	Linked Data Cryptographic Suite Registry. [online] Available at: https://w3c-ccg.github.io/ld-cryptosuite-registry/#bbs-signature-2020 [Accessed 11 Nov. 2023].
[REF-167]	Roots of Trust. [online] Available at: https://csrc.nist.gov/projects/hardware-roots-of-trust [Accessed 11 Nov. 2023].
[REF-168]	Ehret et al. “Reconfigurable Hardware Root-of-Trust for Secure Edge Processing.” 2021 IEEE High Performance Extreme Computing Conference (HPEC), pp. 1–7, doi: 10.1109/HPEC49654.2021.9622830.
[REF-169]	BYOTEE: Towards Building Your Own Trusted Execution Environments Using FPGA. Md Armanuzzaman and Ziming Zhao CactiLab, University at Buffalo https://arxiv.org/pdf/2203.04214.pdf
[REF-170]	Costan V., et al. “Sanctum: minimal hardware extensions for strong software isolation”. SEC'16: Proceedings of the 25th USENIX Conference on Security SymposiumAugust 2016 Pages 857–874
[REF-171]	Keystone Basics. [online] Available at: https://docs.keystone-enclave.org/en/dev/Getting-Started/How-Keystone-Works/Keystone-Basics.html [Accessed 11 Nov. 2023].
[REF-172]	Root of Trust Definitions and Requirements. [online] Available at: https://globalplatform.org/wp-content/uploads/2018/07/GP_RoT_Definitions_and_Requirements_v1.1_PublicRelease-2018-06-28.pdf [Accessed 11 Nov. 2023].
[REF-173]	Keystone Enclave An Open-Source Secure Enclave for RISC-V. [online] Available at: https://riscv.org/wp-content/uploads/2018/12/Keystone-Enclave-An-Open-Source-Secure-Enclave-for-RISC-V.pdf [Accessed 11 Nov. 2023].
[REF-174]	Dayeol Lee, David Kohlbrenner, Shweta Shinde, Krste Asanović, and Dawn Song. 2020. Keystone: an open framework for architecting trusted execution environments. In Proceedings of the Fifteenth European Conference on Computer Systems (EuroSys '20). Association for Computing Machinery, New York, NY, USA, Article 38, 1–16. https://doi.org/10.1145/3342195.3387532
[REF-175]	I. Lebedev, K. Hogan and S. Devadas, "Invited Paper: Secure Boot and Remote Attestation in the Sanctum Processor," *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*, Oxford, UK, 2018, pp. 46-60, doi: 10.1109/CSF.2018.00011.
[REF-176]	Halder, S., Ghosal, A. and Conti, M., 2020. Secure over-the-air software updates in connected vehicles: A survey. *Computer Networks*, *178*, p.107343.
[REF-177]	Mckenna, D. and Automotive, B. (n.d.). Available at: https://www.nxp.com/docs/en/white-paper/Making-Full-Vehicle-OTA-Updates-Reality-WP.pdf.
[REF-178]	Autosar.org. (2023). Standards AUTOSAR. [online] Available at: https://www.autosar.org/standards.
[REF-179]	Etas.com. (2023). AUTOSAR security: Achieving integrated cybersecurity with the Adaptive Platform. [online] Available at: https://www.etas.com/en/company/news-autosar-security-achieving-integrated-cybersecurity-with-the-adaptive-platform.php [Accessed 8 Sep. 2023].
[REF-180]	K. Lemke, C. Paar, M. Wolf, Embedded security in cars (Springer-Verlag, Berlin Heidelberg, 2006), pp. 3–12
[REF-181]	Garrett Motion. (n.d.). Webinar - Stay Ahead of Hackers with Garrett’s Adaptable, Multi-Network Vehicle Intrusion Detection System. [online] Available at: https://www.garrettmotion.com/knowledge-center-category/oem/webinar-stay-ahead-of-hackers-with-garrett-vehicle-intrusion-detection-system/ [Accessed 7 Sep. 2023].
[REF-182]	Buckl, C., Camek, A., Kainz, G., Simon, C., Mercep, L., Stähle, H. and Knoll, A., 2012, March. The software car: Building ICT architectures for future electric vehicles. In *2012 IEEE International Electric Vehicle Conference* (pp. 1-8). IEEE.
[REF-183]	O. Y. Al-Jarrah, C. Maple, M. Dianati, D. Oxtoby and A. Mouzakitis, "Intrusion Detection Systems for Intra-Vehicle Networks: A Review," in *IEEE Access*, vol. 7, pp. 21266-21289, 2019, doi: 10.1109/ACCESS.2019.2894183
[REF-184]	E/E architectures: Optimize their development for automotive projects (techdesignforums.com)
[REF-185]	Bandur, V., Selim, G., Pantelic, V. and Lawford, M., 2021. Making the case for centralized automotive E/E architectures. *IEEE Transactions on Vehicular Technology*, *70*(2), pp.1230-1245.
[REF-186]	Automotive Over-The-Air (OTA) Market Overview. [online] Available at: https://www.marketresearchfuture.com/reports/automotive-over-the-air-updates-market-7606 [Accessed 11 Nov. 2023].
[REF-187]	Mckenna, D. and Automotive, B. (n.d.). Available at: https://www.nxp.com/docs/en/white-paper/Making-Full-Vehicle-OTA-Updates-Reality-WP.pdf.

[REF-188]	Dominic, D., Chhawri, S., Eustice, R.M., Ma, D. and Weimerskirch, A., 2016, October. Risk assessment for cooperative automated driving. In Proceedings of the 2nd ACM workshop on cyber-physical systems security and privacy (pp. 47-58)
[REF-189]	Upstream security global automotive cybersecurity report 2021. Upstream Security Ltd.
[REF-190]	zephyr. [online] Available at: https://zephyrproject.org [Accessed 11 Nov. 2023].
[REF-191]	MCUboot. [online] Available at: https://www.trustedfirmware.org/projects/mcuboot/index.html [Accessed 11 Nov. 2023].
[REF-192]	Open Space Data Link Protocol. [online] Available at: https://gitlab.com/librespacefoundation/osdlp [Accessed 11 Nov. 2023].
[REF-193]	CCSDS Space Packet Protocol, CCSDS 133.0-B-1
[REF-194]	CCSDS TM Space Data Link Protocol, CCSDS 132.0-B-2
[REF-195]	CCSDS TC Space Data Link Protocol, CCSDS 232.0-B-3
[REF-196]	CCSDS Communications Operation Procedure-1, CCSDS 232.1-B-2
[REF-197]	Lee, J. D., & See, K. A. (2004). Trust in Automation: Designing for Appropriate Reliance. Human Factors, 46(1), 50–80. https://doi.org/10.1518/hfes.46.1.50_30392
[REF-198]	ISO/TC 307/JWG 4 "Joint ISO/TC 307 - ISO/IEC JTC 1/SC 27 WG: Blockchain and distributed ledger technologies and IT Security techniques"
[REF-199]	Zero Trust Architecture, NIST Tech Report, SP 800-207 (DOI)
[REF-200]	Software Grand Exposure: SGX Cache Attacks Are Practical, F. Brasser, U. Müller, A. Dmitrienko, K. Kostiainen, S. Capkun, A.R. Sadeghi, USENIX Workshop on Offensive Technologies, 2017
[REF-201]	Preventing Page Faults from Telling Your Secrets, S. Shinde, Z. Leong Chua, V. Narayanan, P. Saxena, AsiaCCS, pp 317-328, 2016
[REF-202]	SgxPectre: Stealing Intel Secrets from SGX Enclaves Via Speculative Execution, G. Chen, S. Chen, Y. Xiao, Y. Zhang, Z. Lin, T-H. Lai, EuroS&P, pp 142-157, 2019
[REF-203]	Dayeol Lee, David Kohlbrenner, Shweta Shinde, Krste Asanović, and Dawn Song. 2020. Keystone: An Open Framework for Architecting Trusted Execution Environments. In Fifteenth European Conference on Computer Systems (EuroSys '20), April 27–30, 2020, Heraklion, Greece. ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3342195.3387532
[REF-204]	Vuillermoz Simone. Analysis of TEE technologies as trust anchors Master Degree Thesis 2021-2022 https://webthesis.biblio.polito.it/secure/25615/1/tesi.pdf
[REF-205]	Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F. Wenisch, Yuval Yarom and Raoul Strackx. 2018. Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution. In USENIX Security.
[REF-206]	Paul Kocher, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. 2019. Spectre Attacks: Exploiting Speculative Execution. In IEEE S&P
[REF-207]	Thomas Bourgeat, Ilia A. Lebedev, Andrew Wright, Sizhuo Zhang, Arvind, and Srinivas Devadas. 2019. MI6: Secure Enclaves in a Speculative Out-of-Order Processor. In MICRO.
[REF-208]	M. Yan, J. Choi, D. Skarlatos, A. Morrison, C. Fletcher, and J. Torrellas. 2018. InvisiSpec: Making Speculative Execution Invisible in the Cache Hierarchy. In MICRO.
[REF-209]	Meni Orenbach, Yan Michalevsky, Christof Fetzer, and Mark Silberstein. 2019. CoSMIX: A Compiler-based System for Secure Memory Instrumentation and Execution in Enclaves. In ATC.
[REF-210]	Rongzhen Cui et al., Emilia: Catching Iago in Legacy Code. Network and Distributed Systems Security (NDSS) Symposium 2021, 21-24 February 2021 ISBN 1-891562-66-5 https://dx.doi.org/10.14722/ndss.2021.24328 www.ndss-symposium.org
[REF-211]	Shweta Shinde, Shengi Wang, Pinghai Yuan, Aquinas Hobor, Abhik Roychoudhury, and Prateek Saxena. 2020. BesFS: A POSIX Filesystem for Enclaves with a Mechanized Safety Proof. In USENIX Security
[REF-212]	Andrew Ferraiuolo, Andrew Baumann, Chris Hawblitzel, and Bryan Parno. 2017. Komodo: Using verification to disentangle secure-enclave hardware from software. In SOSP.
[REF-213]	Qian Ge, Yuval Yarom, David Cock, and Gernot Heiser. 2018. A survey of microarchitectural timing attacks and countermeasures on contemporary hardware. Journal of Cryptographic Engineering (2018).
[REF-214]	Moein Ghaniyoun, Kristin Barber, Yuan Xiao, Yinqian Zhang, and Radu Teodorescu. 2023. TEESec: Pre-Silicon Vulnerability Discovery for Trusted Execution Environments. In Proceedings of the 50th Annual International symposium on Computer Architecture (ISCA '23), June 17–21, 2023, Orlando, FL, USA. ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3579371.3589070
[REF-215]	Lee, J. D., & See, K. A. (2004). Trust in Automation: Designing for Appropriate Reliance. Human Factors, 46(1), 50–80. https://doi.org/10.1518/hfes.46.1.50_30392
[REF-216]	Akl, SG, Taylor PD (1983) Cryptographic solution to a problem of access control in a hierarchy. ACM

	Trans Comput Syst 1(3):239–248.
[REF-217]	Kumar, N., Mathuria, A. Comprehensive evaluation of key management hierarchies for outsourced data. Cybersecur 2, 8 (2019). https://doi.org/10.1186/s42400-019-0026-y
[REF-218]	Data-Sealing. [online] Available at: https://docs.keystone-enclave.org/en/dev/Keystone-Applications/Data-Sealing.html [Accessed 11 Nov. 2023].
[REF-219]	Attestation. [online] Available at: https://docs.keystone-enclave.org/en/dev/Keystone-Applications/Attestation.html [Accessed 11 Nov. 2023].
[REF-220]	Ed25519: high-speed high-security signatures. [online] Available at: https://ed25519.cr.yp.to/ [Accessed 11 Nov. 2023].
[REF-221]	Matheu Garcia, S., Molina Zarca, A., Hernandez Ramos, J., Bernal Bernabe, J. and Skarmeta, A., Enforcing behavioural profiles through Software-Defined Networks in the Industrial Internet of Things, APPLIED SCIENCES-BASEL, ISSN 2076-3417, 9 (21), 2019, p. 4576, JRC117341. https://publications.jrc.ec.europa.eu/repository/handle/111111111/58255
[REF-222]	Maesa, Damiano Di Francesco, et al. "Self-sovereign and blockchain based access control: Supporting attributes privacy with zero knowledge." Journal of Network and Computer Applications 212 (2023): 103577.