An Architecture for Dynamic Trust Management in IoT Security

Stef Verreydt
DistriNet, KU Leuven

Dimitri Van Landuyt
DistriNet/LIRIS, KU Leuven

Michail Bampatsikos
University of Piraeus

Rustem Dautov
SINTEF

Hui Song
SINTEF

Shukun Tokas
SINTEF

Tom Van Eyck
DistriNet, KU Leuven

Sam Michiels
DistriNet, KU Leuven

Apostolis Zarras
University of Piraeus

Thodoris Ioannidis
University of Piraeus

Christos Xenakis
University of Piraeus

Danny Hughes
DistriNet, KU Leuven

Wouter Joosen
DistriNet, KU Leuven

Abstract

Contemporary IoT applications are faced with unprecedented security challenges: they often operate in untrusted or adversarial environments, and the large number of devices creates management complexity and amplifies attack surfaces that are easily beyond manual control. These challenges are exacerbated by limited computational capabilities of some devices and the sheer heterogeneity in terms of technology and capability across device classes.

As traditional security by design approaches focus on introducing and imposing specific countermeasures, these solutions are rigid and static by nature. They are unfit for dealing with the highly dynamic and changing reality in which new technologies continuously emerge, devices migrate, and new attacks and vulnerabilities are discovered at high frequency. In this chapter, we present our joint work on dynamic trust, more specifically, we present a set of mutually reinforcing and integrated technologies that contribute to the run-time assessment of device trust. This notion of trust is highly-context specific, and takes into account (*i*) device capabilities, (*ii*) application context, (*iii*) prior knowledge of devices, and (*iv*) existing security guarantees.

The establishment and the continuous re-assessment of trust scores is a key enabler for building adaptive systems that change their behavior and expectations in function of these trust scores, and this risk-adaptive capability in the end leads to more resilient and secure IoT systems.

1 Introduction

Contemporary IoT applications rely on diverse IoT devices embedded typically in an untrusted environment such as public spaces or within the physical control of users that may be incentivized to attack or game the overall system. For example, smart vehicles can be tampered with by their owners or other malicious actors, which may negatively impact the behavior of a smart traffic application. In other words, the trustworthiness of an IoT application depends on the trustworthiness of the individual IoT devices. In this context, *"trustworthiness"* of an IoT device encompasses several aspects, including trust in the device's identity, trust in its behavior, and trust in the accuracy of the data produced by it. Therefore, measures should be taken to detect devices which are not who they claim to be, behave in unexpected ways, or produce inaccurate data, and to tackle such issues. Furthermore, communications with IoT devices in untrusted environments should also be secured to prevent tampering and information disclosure threats. Finally, as IoT devices are usually heterogeneous in nature, and some may employ

legacy technologies, managing device trust and secure communications becomes even more challenging. This chapter describes the efforts taken during the ERATOSTHENES project (Horizon 2020 research and innovation program under grant agreement No 101020416) to foster trust management throughout the device life cycle.

From the architectural perspective, a trust layer is created in the Edge that hinges upon the interplay between a central Trust Manager and Broker (TMB) that interacts with a Trust Agent (TA) installed on the IoT device. This is shown below in Figure 1.

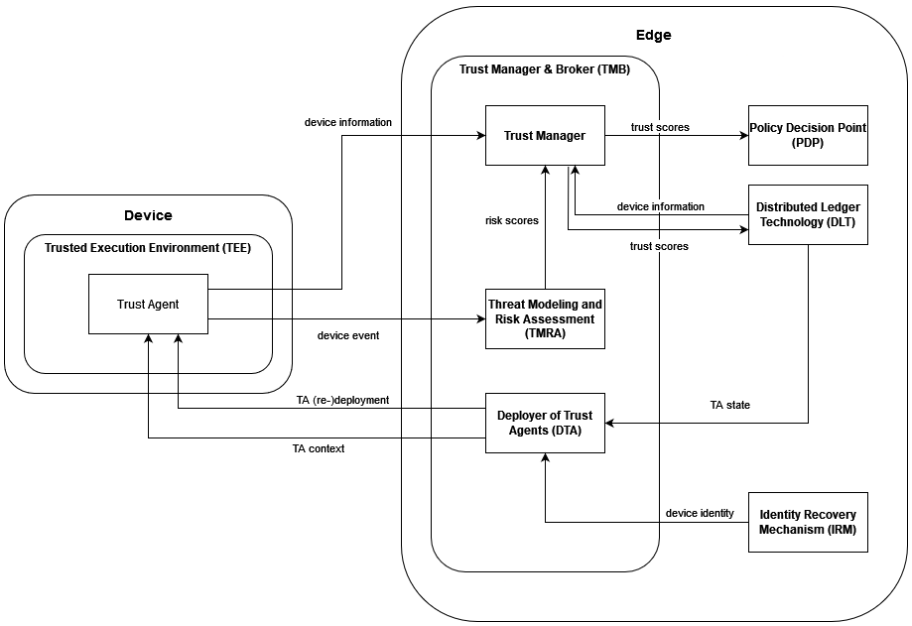

Figure 1: High-level overview of the Trust components and their interaction

The Trust Agent (TA) (depicted on the left) runs on the IoT device and locally collects information such as contextual parameters (e.g., CPU/memory/storage usage, services available and their version numbers, etc.) and relays them to the TMB, either on demand (via a *"pull"* interface), or proactively/recurrently (via a *"push"* interface).

The TMB is a subsystem residing on the computational layer near the device—the Edge—and subscribes to and aggregates this information coming from the devices. It also subscribes to other external sources of information related to individual devices. For example, when a device has migrated from a different domain, prior device information such as trust scores are fetched through the Distributed Ledger (DLT), which is a blockchain-driven information sharing system. Based on all these inputs, trust calculation is performed by the Trust manager, leading to a score that expresses the extent to which the device can be trusted, and this device trust information is made accessible to trust consumers. For example, when the user or the application wants to perform a certain action in the system, the access control system, and more specifically the Policy Decision Point (PDP) may take into account traditional elements such as user attributes but also can base specific access decisions on the trust score of the device.

In the presented solution, these information exchanges between the Trust Agent and the TMB are performed using the MQTT protocol. The fetching of local device context is highly dependent on the nature of the device and the implementation of the TA. The TMB itself shares trust calculation outcomes via MQTT events, but also stores these trust scores in the aforementioned shared ledger (DLT).

The interplay between the device and the TMB is the main point of focus of this chapter. The TMB itself groups a number of technologies and the trust calculation algorithms take into

account different types of security mechanisms that may or may not be at play in the specific context of a device.

Several trust-related services and tools have been incepted, implemented, and integrated in this broader architecture:

- The Threat Modeling and Risk Assessment (TMRA) module performs threat modeling to evaluate specific threat scenarios—specific cases that may involve security issues—in an application-specific manner. The TMRA creates and maintains a digital twin model of the operational systems and performs the automated generation of STRIDE threat scenarios and FAIR-based risk quantification. Risk scores and outcomes are then shared with the TMB.
- The core mechanism described above hinges upon the availability and reliability of Trust Agents on the IoT devices. Technology for the automated (re)deployment of Trust Agents (DTA) on heterogenous devices has been created with specific attention for practical feasibility and scalability of the mechanism towards larger collections of devices (fleets) while dealing with the spurious availability of the devices. Furthermore, to foster the availability and integrity of the Trust Agents, specific support has been created for remote, automatic and secure recovery/roll-back of the TA itself. In specific scenarios where the integrity of the trust agent cannot be fully guaranteed, or when bootstrapping new devices (e.g., upon initial enrolment), this mechanism provides a key secure stepping stone.
- The capability of some device classes to support Trusted Execution Environments (TEE) is an enabler for trust. Trust execution refers to the ability to perform remote attestation, i.e., to remotely verify and ensure the correctness of a computational outcome. Evidently, when such technology is available on devices, it can be used and should be taken into account in trust calculation, as the stronger guarantees of the correct execution on the devices greatly increase the trust in these devices. Specific middleware support has been created to enable secure, remote communication to services running in the TEE, which is crucial to attain the desired level of guarantee.
- The trust mechanism complicates the overall enrollment and bootstrapping process of new devices. Indeed, it may occur that the overall system becomes overly restrictive, i.e., that a new device has not yet provided any enabler for trust, and thus is simply not allowed to perform any action, thus further prohibiting any meaningful use. To alleviate this, and make the overall bootstrapping process practically viable, the entire enrolment process or workflow of a new device has been incepted, designed and implemented for a diverse set of devices, heterogeneous in capability and level of support. The predominant focus in these developments has been to come up with viable strategies to deal with differences in device capability.

Together, these technologies contribute to a new approach of dealing with IoT security, with dynamism and adaptiveness at the core of the trust mechanism.

The remainder of this Chapter provides a more in-depth and technological overview of each of these technical contributions.

2 Trust Broker

2.1 Introduction to the Trust Manager and Broker

Trust is fundamental to IoT security since establishing trust between devices, networks, and users is essential for secure and reliable interoperability in Internet of Things (IoT) ecosystems. More specifically, trust in IoT networks refers to the belief that one IoT device, the Service

Provider (SP), will behave correctly and as expected by prospective Service Requesters (SRs) IoT devices. The enforcement of trust in an artificial society such as IoT is far more difficult, as things do not have an inherent judgmental ability to assess risks and other influencing factors to evaluate trust as humans do. Hence, it is important to quantify "trust" in a manner that can be understood by artificial entities such as IoT devices.

One way to quantify trust is the trust score, a value that reflects a relationship between trustor and trustee, measured by trust metrics, and evaluated by a trust assessment mechanism. Several trust management and evaluation mechanisms focus mainly on security and privacy issues instead of considering the universal meaning of trust and its inherently dynamic nature.

A trust algorithm must calculate a device's trust score, considering various scenarios and trustworthiness-related parameters. To this end, the trust algorithm must be able to collect this information. Due to the high complexity and dynamic of IoT networks with a constantly increasing and changing number of devices, it became apparent that the trust algorithm had to be run on several nodes with high computational resources known as the Trust Manager and Broker (TMB) nodes. This design decision enabled IoT devices to perform their duties without unnecessarily occupying their limited resources by offloading the computation burden to computationally powerful nodes.

With these considerations in mind, the ERATOSTHENES project deemed it necessary to charge the TMB with the following duties:

- To evaluate an IoT device's trustworthiness and compute its trust score.
- To facilitate the exchange of trust-related information among IoT devices and domain services.

To evaluate a device's trustworthiness, the TMB aggregates trust-related information from multiple sources, including the ERATOSTHENES Intrusion Detection System (IDS), Threat Modeling and Risk Assessment (TMRA, discussed in Section 3), and the IoT devices. This results in a comprehensive trust evaluation approach encompassing device-specific characteristics, network threats, and security incidents. Communication between the devices, modules, and the TMB is facilitated through the MQTT protocol, enabling the TMB to calculate a device's trust score.

The MQTT protocol is a lightweight messaging protocol specifically engineered for small sensors and mobile devices. It is optimized for operation over high-latency or unreliable networks, common characteristics of IoT environments.

The MQTT protocol functions through two primary components: the MQTT Broker and the MQTT Clients. The MQTT Broker serves as the server that receives and directs client messages to the relevant destination clients. On the other hand, the MQTT clients are devices or applications that connect to the broker to publish messages, subscribe to specific topics, or engage in both activities.

To guarantee secure communication, the MQTT protocol supports client authentication through usernames and passwords. It also encrypts the data exchanged between clients and the broker using SSL/TLS and implements topic-based access control to manage the permissions related to publishing and subscribing.

An important consideration in the design and implementation of the TMB, particularly the Trust Score Calculation Algorithm, is the diversity of application contexts in which IoT devices operate. The ERATOSTHENES project identifies three main IoT network application contexts: (a) Connected Vehicles, (b) Industry 4.0, and (c) Smart Health. Accordingly, different trust

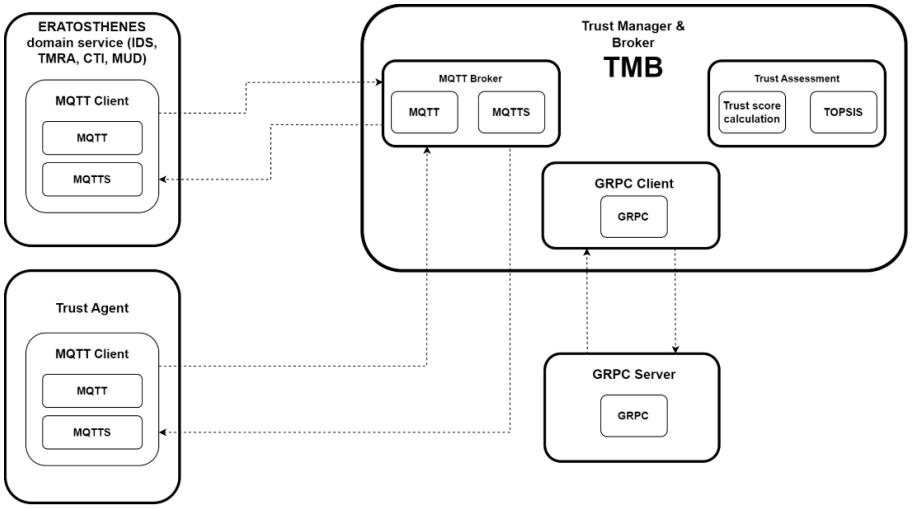

Figure 2: High-level overview of the protocols facilitating the interaction between the TMB, IoT Devices and ERATOSTHENES services

metrics are prioritized within each context, with certain metrics carrying more weight than others in determining a device's overall trust score.

2.2 Device Trustworthiness Evaluation

The evaluation of a device's trustworthiness, as discussed earlier, is influenced by various device-specific parameters (e.g., the device's CPU utilization) as well as assessments derived from ERATOSTHENES services, such as the TMRA and IDS. As a result, assessing a device's trustworthiness becomes a complex multi-attribute problem. Additionally, an SR may need to determine which SP to approach for a specific service. Thus, evaluating device trustworthiness is formulated as a Multi-Attribute Decision Making (MADM) problem [1].

To address this issue, the TMB offers two key functionalities: (a) the computation of a device's trust score and (b) the ranking of devices within the network based on their trustworthiness. The first functionality quantifies a device's trustworthiness, while the second aids prospective SRs in selecting the appropriate SP for the desired service.

Both functionalities rely on a set of device-related parameters (e.g., root of trust, associated cyber risks), along with corresponding weights that reflect the significance of each parameter in determining the device's trustworthiness. The trust score is calculated using a weighted sum function, while the ranking of devices is carried out via the Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) [2], a widely adopted ranking method.

Whenever a significant event occurs for a device (e.g., a software update), the TMB triggers the trust score calculation and TOPSIS functions to reassess the device's trustworthiness. The specific weights used in the weighted sum function and TOPSIS are context-dependent, varying according to the particular application domain in which the IoT device operates. The concept "application context" refers to the type of IoT ecosystem the device is part of, such as Autonomous Vehicles, Industry 4.0-enabled factories, or Smart Medical Devices. Even if certain trust-related parameters remain consistent across different contexts, the weighting of those parameters differs according to the specific ecosystem.

The TMB works as follows. Initially, an IoT device's Trust Agent establishes an MQTT connection to the TMB, and then the former publishes the device's data on an MQTT topic. The broker component of the TMB collects this data. It forwards them to the TMB's internal modules, namely the TMRA, Manufacturer Usage Description (MUD) manager, IDS, and the Cyber

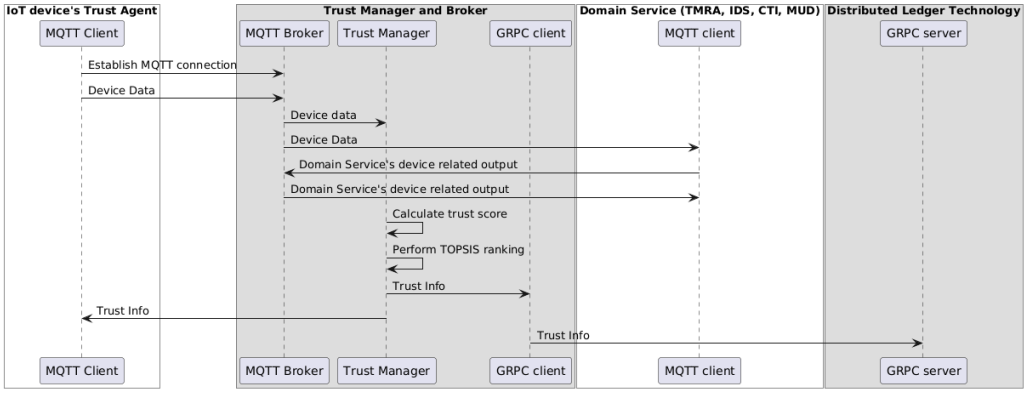

Figure 3: Trust score calculation sequence diagram

Threat Intelligence (CTI) manager, which are also MQTT clients. These modules process the device data and send their outputs to the MQTT broker, who forwards them to the Trust Manager functionality of the TMB or other MQTT clients. Then, the Trust Manager invokes the trust score calculation function and the TOPSIS implementations to conduct the device trust assessment. Once the trust score is calculated, the TMB invokes a gRPC Remote Procedure Call through its gRPC client to the DLT's gRPC server. Through this call, it writes the trust score to the DLT while it informs the device via an MQTT message about its corresponding trust score. Figure 2 provides a high-level overview of the interactions between the TMB, the devices, and the other services of the ERATOSTHENES architecture. Furthermore, it depicts the components that make up the TMB. The operational flow of a device's trust score calculation is shown in the sequence diagram depicted in Figure 3.

The trust evaluation is event-triggered; these events are messages transmitted over MQTT topics. Table 1 describes these events and their corresponding topics.

Table 1: MQTT events that trigger trust evaluation

Event	Topic	Payload/Description
New Device	entity/entityID/new	Device (entity) type and parameters (A new device sends its trust related information)
Device Removed	entity/entityID/remove	An announcement to remove a device from the network
Device Data	device/entityID/data	Device Data (through this topic, the Trust Agent sends the Device Data to the TMB)
Trust Score Calculation	entity/entityID/agent/calculate	Initial or updated trust score for a device (the TMB calculates the trust score of a device with a specific ID and forwards it to the subscribers of that topic)
Risk Score Calculation	entity/entityID/tmra/riskscore	A risk score computed by the TMRA module (through this topic, the TMB will receive the risk score from the TMRA module for the device with the specified ID)
Interaction with IDS	entity/entityID/ids	IDS monitoring score (through this topic, the TMB interacts with the IDS module to obtain an IDS monitoring score)
Interaction with CTI	entity/entityID/cti	Through this topic the CTI interacts with the TMB and sends threat related information
Interaction with MUD	entity/entityID/mud/MUDfilejson	Through this topic the MUD interacts with the TMB and sends the MUDfilejson file which contains manufacturer specification and expected behavior for a device with a specific ID

2.3 Summary

In summary, the TMB plays a vital role in the ERATOSTHENES ecosystem's architecture since it acts as the enabler of communication between the TMRA, the IoT devices, the MUD manager, the CTI agent and the IDS. Moreover, it facilitates the formation of trust relationships between entities in the IoT network as well as evaluates the trustworthiness of IoT devices and

manages trust records for every IoT device. The most prominent feature of the TMB is that it considers multiple aspects related to an IoT device before evaluating its trustworthiness. Finally, by facilitating the transmission of trust information through the DLT the TMB plays a crucial role in forming trust relationships among different domains.

3 Threat Modeling and Risk Assessment

Trust in an IoT system entails multiple aspects: trust in the identity of a device, trust in it behaving according to expectations, trust in the validity of the messages it sends, and so on. Assessing these aspects requires considering potential security threats and their associated risk. For example, an IoT device that implements outdated authentication mechanisms is more likely to be spoofed, and unencrypted data flows are easily tampered with, which should be taken into account when calculating trust scores.

One way to identify security threats and estimate their risk is threat modeling, which involves the up-front investigation of common security threats in a system's design or architecture, both in terms of whether they would be feasible (likelihood) and the consequential harm (impact). Systematic, system-centric approaches act upon an abstraction of the system under design. This abstraction usually takes the form of a Data Flow Diagram (DFD), which defines the envisioned or anticipated data flows between processes, (external) entities and data stores. An example DFD for a connected vehicles system is depicted in Figure 4. Based on this DFD, potential security and privacy threats can be identified by iterating over all elements, for example that vehicles can be spoofed, and data flows can be tampered with.

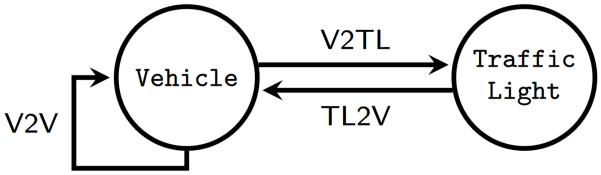

Figure 4: Example DFD for a connected vehicles application

Numerous threat modeling methodologies and tools exist to automate (parts of) this workflow [3], for example through automated threat elicitation or risk calculation. All of these, however, consider risk at the design level, without considering that the risk for specific entities or instances of design-level entities may differ considerably based on certain parameters or the specific context. For example, spoofing a priority vehicle is potentially more severe compared to spoofing a non-priority one, and different vehicles may implement different authentication methods, some of which may be stronger than others. The lack of expressivity in existing threat modeling approaches has shown to be especially problematic in architectural styles such as peer-to-peer or decentralized systems in which the modelled system elements are not singular nor centrally governed. For example, an earlier study has identified this particular issue hindering the threat analysis of distributed ledger applications [4]. While such approaches allow to assess risk at the basis of class-level elements, they fail to consider and address risk factors that emerge at instance level. This, in turn, hinders device-specific trust calculation taking into account security risk factors, for example for the specific vehicles depicted in Figure 5.

3.1 Instance-Centric Threat Modeling

To tackle this issue, and allow risk assessments for specific IoT devices, the Threat Modeling and Risk Assessment (TMRA) module implements a novel threat modeling framework called "instance-centric threat modeling" [5] which enables specifying and analyzing both the high-level

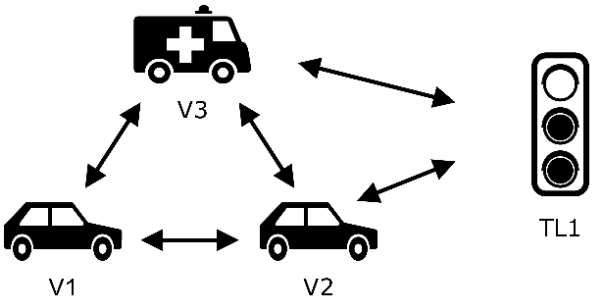

Figure 5: Example of a specific connected vehicles scenario [5]

design of a system (e.g., Figure 4) as well as specific instances of entities specified in the design (e.g., Figure 5). Specifically, the main goals of the framework approach are to enable structured modeling of instance-level situations and configurations, and in turn to support more precise risk calculation based on the additional, instance-level information. A high-level overview of the proposed threat modeling approach is shown in Figure 6.

In short, the instance-centric threat modeling framework enables threat modelers to capture the general structure and context of the system by creating the design model, and manually explore different (potential) instance-level scenarios by modeling one or more instance models. For example, if a design-level analysis reveals that spoofing vehicles remains a high-risk threat even

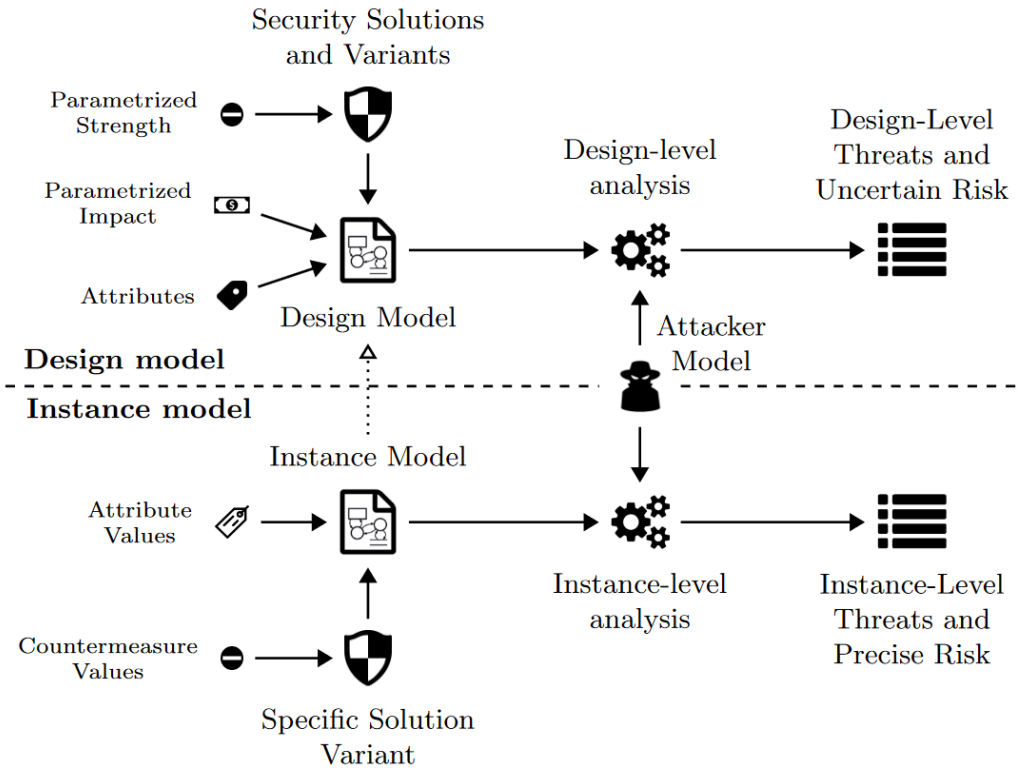

Figure 6: High-level overview of instance-centric threat modeling. The design model (e.g., Figure 4) specifies the high-level structure of a system and can be analyzed to reveal major flaws in the design, but the resulting risk values will be inherently uncertain as the design model covers all potential configurations and scenarios. One or more instance models (e.g., Figure 5) can be created to model specific instances of the entities in the design model, with concrete parameters, which allows for more precise risk calculation [6]

though vehicles are authenticated, threat modelers might try to identify in which situations spoofing threats are more likely and/or impactful by systematically modeling and analyzing scenarios. We refer to the original paper [5] for additional details.

3.2 Digital Twin Threat Modeling

The TMRA leverages the proposed instance-centric threat modeling approach in a more automated manner by capturing and maintaining an instance model which represents the digital twin of an operational system [6]. Specifically, the design model captures types of entities (i.e., IoT devices), for smart vehicles and traffic lights, and the digital twin captures instances of these devices, for example several specific vehicles with unique parameters. This digital twin can be updated continuously based on real-world events, allowing for accurate and real-time re-evaluation of threats and risk. The outcome of such threat analysis is then be leveraged during operations by the trust manager (Figure 1) to calculate trust metrics taking into account the likelihood of devices being spoofed. Figure 7 illustrates this approach which involves synchronizing the digital twin model at the basis of real-world events and then leveraging the risk calculation outcomes in a risk-adaptive security context.

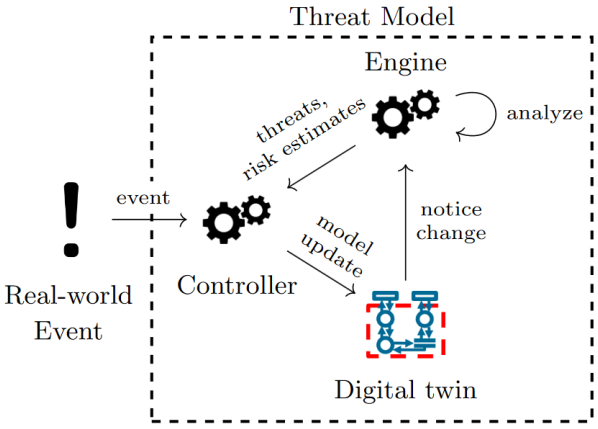

Figure 7: Conceptual overview of the digital twin threat model as implemented by TMRA [6]

The TMRA consists of three components: the digital twin model is an abstract representation of the real-world system, the engine re-analyses the digital twin when it is updated, and the controller interprets real-world events and updates the digital twin model accordingly. Events and analysis results are relayed through a message broker. We shortly summarize each of these components in what follows.

Table 2: Real-world events relevant to the TMRA, and the corresponding MQTT topics

Event	Topic	Payload
New Device	entity/entityID/new	Entity type (e.g., "Vehicle") and parameters
Device Removed	entity/entityID/remove	
New Communication	dataflow/entityID/new	SenderID;RecipientID
Parameter Changed	entity/entityID/atr/atrname	New parameter value

Real world events are relayed to the TMRA via MQTT. This could either be done by the devices themselves (e.g., vehicles announcing that they approach a traffic light), or by some observer which detects devices and sends corresponding events to the message broker. Relevant events for the threat model include (*i*) entities entering a system (e.g., a vehicle approaching a traffic light), (*ii*) entities being removed from a system (a vehicle driving away from a traffic light), (*iii*) entities interacting with one another (vehicles communicating with one another), and (*iv*) entities being updated (vehicles re-authenticating). A generic, context-agnostic MQTT topic is defined for each of the above-mentioned events, as shown in Table 2. Updates to the

threat model are also published to the message broker, allowing other components to subscribe to threat or risk updates. The specific output topics are shown in Table 3. Interested parties (for example, the trust manager in Figure 1) can subscribe to all risk updates or to fine-grained events, for example the risk of spoofing for a specific vehicle.

Table 3: MQTT topics for output risk values

Output	Topic	Payload
Overall Risk	entity/entityID/risksore	Risk value
Specific Threat Type	entity/entityID/threatType/riskscore	Risk value

The *digital twin* model is a modelled abstraction of a real-world system and corresponds to an instance model. For example, for the smart traffic case, this would be the collection of specific traffic lights and vehicles, and instances of data flows occurring between them. The digital twin model also captures attribute values (e.g., whether or not the vehicle is a priority vehicle) and security solutions (e.g., which authentication protocol is used by each vehicle, and what is the elapsed time since they last authenticated).

The *engine* is an extended version of SPARTA[1] which runs as a service listening for changes to the digital twin and automatically re-analyses the model if any changes are detected. It elicits threats and calculates risk estimates for each threat, taking into account security solutions (e.g., authentication) and attributes (priority, time since last authentication). Furthermore, it keeps track of all identified threats and their risk scores and can be queried to request the latest updates.

The *controller* parses MQTT messages, updates the digital twin model accordingly, and publishes the updated threats and risk estimates to the message broker. During initialization, the controller subscribes to the topics in Table 2. This is done through wildcards to allow subscribing to multiple topics at once. For example, the topic "entity/ + /new" matches with any topic where the wildcard ('+') is replaced with an arbitrary string, so the controller is notified when any new entity enters the system. As described, the MQTT topics are generic, and need to be translated to application-specific model updates. For example, if the controller receives a message with topic "entity/V 1/new" and payload "Vehicle", then a new vehicle should be created and added to the digital twin. Updates to the digital twin may induce new threats or change risk scores, as identified by the engine. The controller tracks the threats and risk scores in the engine to be notified of any updates to the threat landscape. Furthermore, the controller publishes such updates to the MQTT broker.

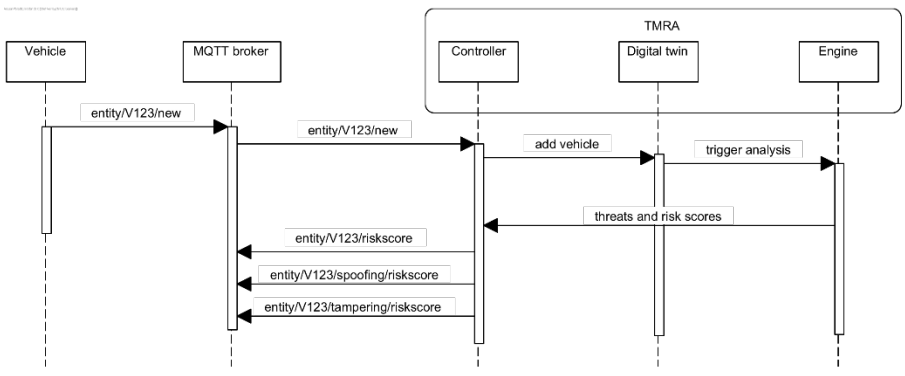

Figure 8: Sequence diagram for vehicles announcing their presence to a traffic light

Figure 8 illustrates the workflow for a smart traffic system where vehicles announce their presence as they approach a traffic light. Such system could be used to dynamically regulate traffic lights

[1] `https://sparta.distrinet-research.be/`

and minimize the amount of time vehicles are waiting at an intersection. Furthermore, priority vehicles such as ambulances approaching a traffic light may request immediate actions from a traffic light in case of emergency. In this context, dynamically managing trust is critical to avoid actors from maliciously influencing the system such that they can move along traffic as fast as possible, potentially at the expense of other vehicles. Specific threats include attackers claiming to be priority vehicles, sending messages from non-existent vehicles to make the queue appear longer, and intercepting or tampering with messages sent from other vehicles.

To illustrate how threats and risk scores support dynamic decision making in an IoT system, consider the scenario where a vehicle claims to be a priority vehicle, and requests immediate right of way at an intersection. When such vehicle announces its presence, the threat model is updated by adding an entity to the system model, and the new model is analysed, resulting in new threats (e.g., the vehicle being spoofed), as well as risk scores for each threat. Depending on the specific situation, different actions could be taken by the system:

- If the vehicle implements state-of-the-art authentication protocols, and can thus provide strong evidence for its identity, the likelihood of that vehicle being spoofed is low, resulting in a lower risk score and, in turn, a higher trust score, which will most likely result in immediate actions from the traffic light.
- If the vehicle cannot provide strong evidence for its identity, the likelihood of spoofing is higher, which may lead to the system ignoring the request, or to the system requesting the vehicle to re-authenticate. In the latter case, re-authentication induces changes to the digital twin model and updates to the threats and risk scores calculated by the TMRA, and thus also updates to the trust scores calculated by the TMB, which in turn may change lead to a different decision by the system.

3.3 Summary

In summary, by continuously updating a digital twin threat model, the TMRA dynamically identifies new threats and calculates updates risk scores based on the specific situation. In the ERATOSTHENES project, these outputs are leveraged by the trust manager to calculate trust scores for IoT devices, which requires taking into account the likelihood and potential impact of certain security threats such as devices being spoofed, or messages being tampered with. More in general, the TMRA enables self-adaptive security controls, for example by enforcing stronger countermeasures when the overall risk of spoofing exceeds a certain threshold, fine-grained and informed access control based on specific device parameters, and other run-time policy enforcements.

4 Automated Deployment and Recovery of Trust Agents

A Trust Agent (TA) is a software component deployed on IoT devices within the ERATOSTHENES ecosystem to collect run-time information used for assessing the trustworthiness of the device. It collects trust-related data, such as hardware resource usage, installed software, and security metrics. This information is then used by the Trust Manager and Broker (TMB) to ensure that the device's behavior aligns with predefined trust parameters. The TA plays an important role in maintaining the security, integrity, and reliability of IoT devices, and TA lifecycle management is an important challenge in this context. Throughout its lifecycle, a TA may undergo multiple updates triggered by various factors, such as security vulnerabilities, software bugs, or performance enhancements. By aligning the TA lifecycle with secure software update practices, the ERATOSTHENES project ensures that the IoT devices in its ecosystem remain secure, reliable, and trustworthy.

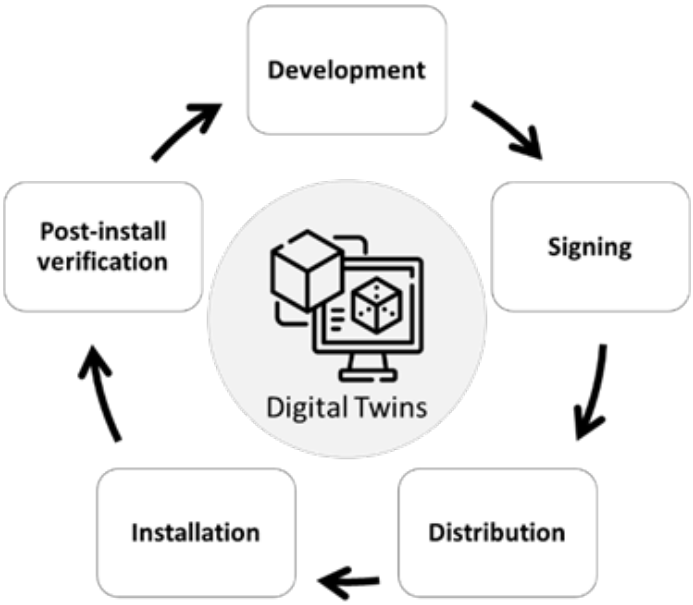

Figure 9: Trust Agent lifecycle supported by the Digital Twin platform

Please note that this section focuses only on managing the TA lifecycle as part of the TMB functionality. From a broader perspective, a TA can essentially be seen as a software component, which itself can decrease or increase the level of device trustworthiness depending on the presence of vulnerabilities, applied security patches, execution traces, etc. From this perspective, supporting the lifecycle of any software deployed and running on IoT devices is another important instrument for IoT trust management in ERATOSTHENES, which is out of the scope of this chapter.

4.1 Trust Agent Lifecycle

The lifecycle of a TA in the ERATOSTHENES ecosystem follows the Trusted Computing Group's (TCG) "Guidance for Secure Update of Software and Firmware on Embedded Systems"[2] and includes five main steps: secure development, secure signing, robust distribution, secure installation, and post-installation verification. This lifecycle, depicted in Figure 9, is closely tied to the lifecycle of the IoT device itself, merging when the device is initially enrolled into the trusted ecosystem. At this point, the IoT device receives its initial TA, marking the start of its trusted operations. From this moment, the TA lifecycle becomes intertwined with the IoT device lifecycle, as the device will continually receive updates to its TA during its operational lifespan.

1. ***Secure Development:*** The secure development of TAs is expected to be implemented by IoT device manufacturers or trusted third-party developers following strict security standards. TAs are designed with security in mind, ensuring they do not introduce any additional threats themselves. This includes securing the software from the ground up, considering the potential attack vectors in the IoT environment. It is also envisioned by ERATOSTHENES that the newly released TA versions will contain patches to certain vulnerabilities identified as a result of continuous trust monitoring and assessment. The development process also ensures that the TA can function across various hardware and software configurations typical in IoT systems, like edge gateways and personal health monitoring devices used in the Smart Healthcare pilot.

[2] `https://trustedcomputinggroup.org/resource/tcg-guidance-for-secure-update-of-software-and-firmware-on-embedded-systems/`

2. ***Secure Signing:*** Once developed, the TA is cryptographically signed, ensuring its integrity and authenticity. In ERATOSTHENES, we assume that device manufacturers generate and manage a pair of cryptographic keys: a private key for signing TA updates and a corresponding public key for verifying them. The private key is stored securely to prevent unauthorized access and potential misuse. On the other hand, the public key is embedded within the IoT devices during the manufacturing process, enabling these devices to verify the authenticity of received updates at a later stage. The signing mechanism guarantees that any TA received by an IoT device can be trusted as originating from an authorized source and has not been tampered with. This step ensures that only verified TAs are allowed to be distributed and installed within the managed IoT ecosystem.

3. ***Robust Distribution:*** In ERATOSTHENES, the robust distribution phase is one of the key innovations, ensuring that TAs are deployed across a distributed network of IoT devices securely and efficiently. Inspired by the model-driven engineering techniques [7, 8], this process is facilitated by the use of Digital Twins (DTs), which represent each IoT device in the virtual space, reflecting the real-time status and context of each device [9]. The DT platform listens for new updated contextual information collected by device-side monitoring agents, and then uses this cyber-physical-social context of each device to ensure that the correct TA is assigned. This means that various factors such as hardware capabilities, network conditions, and the operational context of the IoT device are considered during the assignment of TAs, making sure that the right TA is deployed for each device. A key part of this phase is the asynchronous installation, enabled by the Desired-Reported Property pattern of DTs. The system first applies updates to the DT, where the desired state of the TA is written to the corresponding property. This desired state is then continuously communicated to the real physical device. Noteworthy, the device might be temporarily offline, but as soon as it gets back online it eventually receives its desired state. This pattern supports a scalable approach, allowing updates to occur even across large fleets of devices, potentially numbering in the hundreds or thousands. This way, DTs also help in scalability, as they allow ERATOSTHENES to manage updates across large fleets of devices in parallel. The use of publish-subscribe communication via MQTT ensures efficient and timely distribution of updates, while the DTs enable real-time tracking and coordination, making the system robust enough to handle a large number of devices in varying states of readiness for updates.

4. ***Secure Installation:*** After robust distribution, the secure installation of TAs ensures that the update is applied safely to the IoT device. This is implemented using device- and platform-specific adapters instantiated for each device (or a group of functionally identical devices) [10]. Mostly, at this step ERATOSTHENES relies on the existing tools for software installation over the network. Some common examples include adapters for interacting with the Docker Engine API, over SSH, or using a RESTful API. The IoT device then reports back with its reported state after installation, creating a smooth update process that does not require synchronous communication.

5. ***Post-installation Verification:*** The final phase of the TA lifecycle involves verifying that the newly installed TA is functioning as expected. In ERATOSTHENES, this step involves confirming that the TA is successfully running on the IoT device and is providing accurate trust metrics. This is verified through the DT, which compares the desired state (the expected TA version) with the reported state (the actual state of the device post-installation). If the reported state matches the desired state, the TA is considered successfully installed. If discrepancies are detected, the system can initiate recovery or roll-back procedures.

4.2 Trust Agent Updates, Roll-back, and Recovery

TA lifecycle management in the ERATOSTHENES ecosystem is a threefold process, encompassing actual updates, roll-back, and recovery. Each of these processes is designed to maintain the security, stability, and functionality of TAs and, thus, the host IoT devices, ensuring they remain trustworthy throughout their lifecycle.

1. *Actual updates* are the primary type of TA lifecycle activities and are typically triggered in response to identified security vulnerabilities, threats or risks. These updates are released either by the device manufacturer or a trusted third-party software developer to patch known security flaws or improve the performance and functionality of the TA. Such updates are critical for pro-actively maintaining the trustworthiness of IoT devices, especially as new threats or weaknesses are discovered in the broader ecosystem. The update process follows the principles of secure distribution and installation to ensure that the IoT device receives and applies the update in a secure and reliable manner.

2. *Roll-backs* occur when the system needs to revert to a previous stable version of the TA, or even the very initial default version. Roll-backs are usually triggered in response to malfunctioning or operational issues rather than cybersecurity incidents. For instance, if a recently installed TA version causes performance problems or system instability, the system can perform a roll-back to restore the device to a previously stable state. This process ensures that devices continue to function smoothly without being affected by faulty or incompatible updates. Roll-backs are different from recovery in that they generally address software issues rather than failures caused by attacks or significant disruptions. In relation to this, ERATOSTHENES employs a blue-green deployment strategy, where two environments (blue and green) are maintained on each IoT device. The new TA version is first deployed to the green environment for testing and validation. If everything functions as expected, traffic is switched from the blue (previous version) to the green (new version) environment. This seamless transition reduces downtime and ensures a fallback option is available. If any issues arise during installation, the system can roll back to the blue environment, effectively reverting to the previous TA version without causing disruptions.

3. *Recovery* involves the re-installation of the TA along with the recovery of its state from a stored backup, typically in the Distributed Ledger Technology (DLT) system. Unlike roll-backs, recoveries are often triggered by more severe incidents, such as device failures or cyberattacks, where the TA has been compromised or rendered inoperable. In these cases, the system not only reinstalls the TA but also restores its state, including key contextual information and trust metrics, to ensure continuity of service and the integrity of the device. Recoveries are designed to restore the trustworthiness of the device after significant disruptions, which is an important aspect of maintaining IoT security and reliability.

4.3 State Preservation

State preservation is crucial when updating, recovering, or rolling back TAs because it maintains the continuity and integrity of operations within IoT devices. A stateful approach, as implemented in the ERATOSTHENES project, allows the system to maintain important contextual information and trust metrics across these processes. This is essential for IoT ecosystems where the history of device actions, configurations, and security settings play a significant role in determining the trustworthiness and proper functioning of devices.

In contrast to stateless methods, which only focus on reinstating functionality without keeping track of the device's operational history, a stateful approach allows for a more accurate recovery and update process [11]. Stateless updates or recovery would simply re-apply the software without regard for the previously held configuration or operational data, potentially leading to

gaps in trust, security, and performance continuity. For instance, critical settings such as device identity, security logs, or resource usage would be lost during a stateless recovery, potentially making the device vulnerable to security threats or operational inefficiencies. A stateful recovery or update preserves these critical parameters, so that the device resumes operations as they were before the interruption.

In ERATOSTHENES, stateful transitions of TAs are made possible by storing the device and TA states in a DLT system, specifically Hyperledger Fabric. The DLT serves as a tamper-proof, decentralized repository where key contextual information and operational data about each IoT device and its TA are stored as assets. This includes data such as the installed software packages, hardware usage metrics, trust scores, etc. By storing this information in the DLT, it is possible to retrieve it later in a secure and non-mutable manner whenever necessary. During updates, roll-backs, or recoveries, the stored state information is fetched from the DLT and 're-applied' to the newly deployed or restored TA. For instance, in the Smart Healthcare context, this could include retrieving the list of critical software packages installed on a healthcare gateway or past resource usage readings to assess the device's health. Without this stateful transition, devices might return to a default state, losing key operational data and compromising the continuity of service. In this context, the stateful recovery process ensures that devices can not only restore functionality but also regain their previous level of trust and reliability without losing important historical data.

The use of DTs in ERATOSTHENES further facilitates the stateful transition by acting as a *recovery checkpoint* representing the last-known status and context of each device before. When an update or recovery is triggered, the DT helps compare the desired state (what the TA or device should be) with the reported state (what it currently is), ensuring that any state differences are identified and corrected. This state preservation and comparison process serves to bring the IoT device back to a fully operational and trustworthy state after an update or recovery, with its critical operational history intact.

4.4 Summary

Trust agents (TAs) are deployed on the IoT devices and collect run-time information which is used by the TMB to assess the overall trustworthiness of a device. To safeguard the correctness of this run-time information, and thus of the calculated trust scores associated with devices, it is crucial to protect the TA itself from tampering and other security threats. The ERATOSTHENES project therefore defines a secure lifecycle for the TA, as well as strategies for secure roll-backs and recovery, with a particular focus on state preservation to ensure continuity as the TA are updated or recovered.

5 Trusted Execution Environments and Remote Communication

The ERATOSTHENES architecture includes several components deployed on the IoT devices themselves, such as the Trust Agent described in the previous section or an SSI Agent, which are required to assess the trustworthiness of the device. However, IoT devices may be deployed in untrusted environments, and malicious applications may be running on them, so additional protections are needed to guarantee the integrity components on the IoT devices, and the confidentiality of the data they process. The ERATOSTHENES architecture therefore leverages Trusted Execution Environments (TEEs).

A TEE is a set of hardware and software components that realizes a secure area of memory on the main processor (the Secure World) and guarantees that the code and data loaded in the Secure World is protected from untrusted software running in the Normal World. Additionally,

a root of trust[3] ensures that authenticity and integrity hold for the code that is loaded during boot. The TEE can run arbitrary code that can be loaded at boot or at runtime. As such, a TEE offers an isolated computational environment that provides authenticity, integrity and confidentiality of all code running in it as well as data and runtime state of the environment stored inside. An overview is shown in Figure 10.

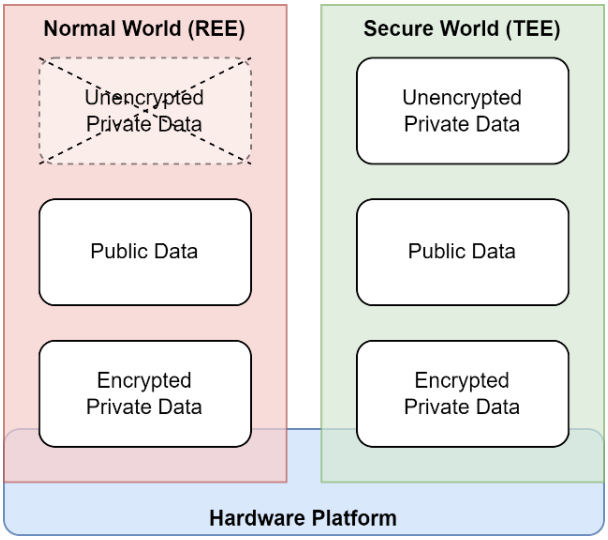

Figure 10: Overview of multiple types of data and their allowed processing on IoT devices

The ERATOSTHENES project provides a generic TEE and supporting services for IoT devices, building upon state-of-the-art technologies such as i.MX6[4], the QEMU emulator[5], Arm TrustZone[6], OP-TEE[7] operating system, the GlobalPlatform Sockets API[8], and Mbed TLS 3.6.0. The TEE acts as a root of trust on IoT devices that can deliver security features throughout an IoT device's software lifecycle, even in the presence of untrusted applications. Two enhancements to existing TEE technologies were made as part of the ERATOSTHENES project, which will be summarized in this section.

5.1 Secure Scheduling and Sharing of Peripherals

While existing TEE technologies provide confidentiality and integrity for resources in the Secure World, they do not protect critical tasks running in the Secure World from Denial of Service (DoS) attacks. Especially in Cyber Physical Systems (CPSs), which interact continuously with the physical world and are hence bound by stringent timing and safety requirements, this can cause serious security and safety hazards, for example by delaying sensor readings. This problem is exacerbated by the fact that CPSs are increasingly connected to the internet, for example with Industry 4.0, where industrial assets such as machines, robots and production lines are connected with real-time analytics and control systems. In practice, CPSs are connected to the internet by using reputable commodity OS components and applications because of their ease of implementation and well-tested nature. Unfortunately, these significantly increase the attack surface due to their large code base, exposing vulnerabilities which range from nuisances to life-threatening malfunctions.

[3] https://csrc.nist.gov/glossary/term/roots_of_trust
[4] https://www.ezurio.com/single-board-computer/nxp-imx6/bd-sl-i-mx6
[5] https://optee.readthedocs.io/en/latest/building/gits/build.html#get-and-build-the-solution
[6] https://www.arm.com/technologies/trustzone-for-cortex-a
[7] https://www.trustedfirmware.org/projects/op-tee/
[8] https://globalplatform.org/specs-library/tee-client-api-specification/

To prevent attackers from tampering with the schedule of real-time tasks or overloading resources, Mr-TEE [12], a novel, hardware-enabled TEE to host and schedule safety-critical code, was developed as part of the ERATOSTHENES project. Concretely, Mr-TEE protects against remote adversaries which may exploit software vulnerabilities to launch DoS attacks on system resources or critical peripherals by executing arbitrary code in the Normal World. An overview of the Mr-TEE architecture is shown in Figure 11. A short summary of the main components is provided below.

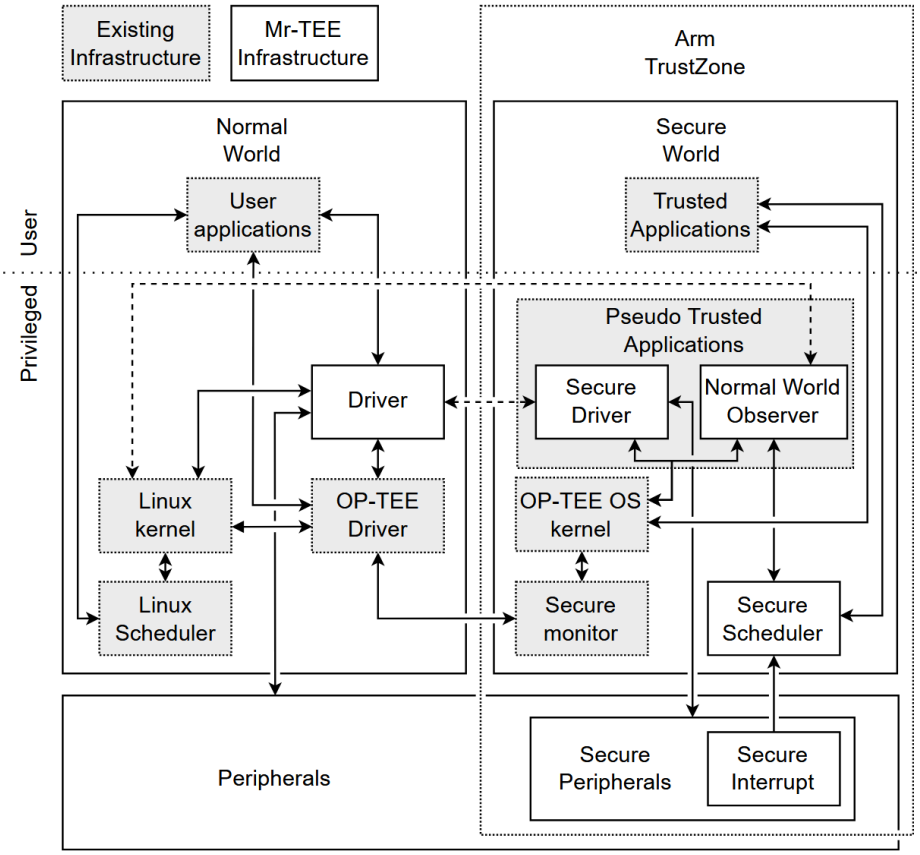

Figure 11: Architecture of Mr-TEE

- ***Secure Scheduler.*** Most existing TEE implementations lack a full-featured scheduler, depending instead on the Normal World scheduler. To provide availability guarantees, we integrate a minimal yet sound real-time scheduler in the Secure World, which makes it possible to schedule critical tasks in real-time independently from the Normal World. This provides a mechanism through which developers can partition computational resources between critical and non-critical functionality.
- ***Secure Peripherals.*** To protect peripherals against bugs or attacks from the (compromised) Normal World, Mr-TEE provides the ability to map certain peripherals to the Secure World. This mapping disables direct access to these peripherals by any Normal World process, forcing all interactions through the Secure World. Naturally, because of the trusted state of the Secure World, applications running in the Secure World can still access peripherals mapped to the Normal World. As the criticality of each peripheral is application-dependent, Mr-TEE gives the designer of the system the choice as to which peripherals are mapped to which world. This ensures the best response time for peripherals that will be exclusively used in only one of the worlds, while minimizing code base size and developer effort by avoiding the need to manage all peripherals in the Secure World.

- ***Shared Peripherals.*** Not all peripherals can be mapped to a single world, as software in both worlds may require access to the same peripheral for normal operation. In this case, Mr-TEE provides a novel mechanism called Shared Secure Peripherals, shown in Figure 2, wherein the Secure World provides a Secure Driver that is responsible not only for providing access to these peripherals for Trusted Applications, but also for providing interfaces via which the Normal World can access the functionality of these same peripherals. The Secure Driver is thus able to control the access of the Normal World to the peripheral, preventing any untrusted applications from blocking peripheral access for critical software running in the TEE.
- ***Normal World Observer.*** Building upon the presence of a real-time scheduler in the Secure World and a notification system to the Normal World, Mr-TEE offers a Normal World Observer, which is capable of periodically checking the running state of the Normal World OS. Whenever scheduled, the Normal World Observer challenges the Normal World kernel to respond in a certain time frame. The fulfilment of the challenge implies that the Normal World is still actively running and is neither frozen nor crashed. Additionally, if the Normal World has crashed, a status report can be obtained through a snapshot of the Normal World registers and memory. In this way, the Secure World can trigger positive action in the case of unexpected behaviour.

In summary, Mr-TEE provides a sound yet minimal real-time scheduler in the Secure World to guarantee safe execution of critical tasks, as well as the means to securely share critical peripherals between the Secure World and the Normal World to guarantee availability yet enable Normal World applications to execute as normal.

5.2 Secure Communication

Next to the TEE itself, which secures resources on IoT devices, the ERATOSTHENES project also provides a generic, end-to-end encrypted communication channel to move private data in or out of the TEE, allowing outside devices and remote servers to initiate connections with those applications. A high-level overview of this communication channel is shown in Figure 12. The main components are the following.

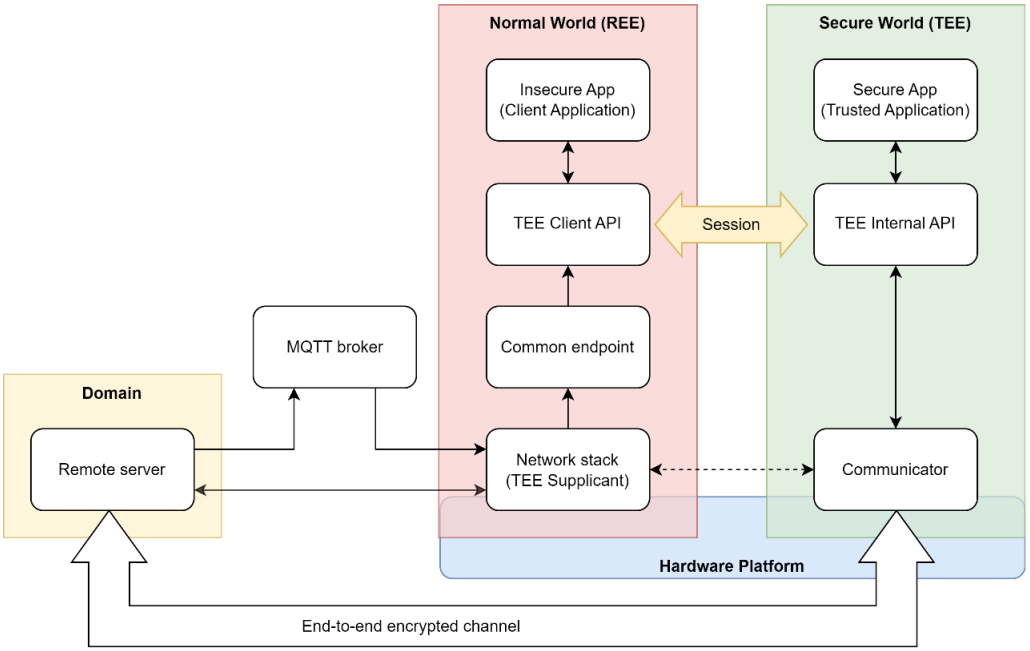

Figure 12: Overview of the end-to-end encrypted communication channel

- The *Communicator* library in the TEE provides a generic interface for trusted applications to initiate end-to-end encrypted communications channels for connections to remote devices or servers.
- Because applications inside the TEE can only be called by applications running in the Normal World, remote devices or servers cannot directly initiate connections. The *Common endpoint* in the Normal world is therefore responsible for listening for connection requests coming from outside the device and forwarding these requests to the trusted applications in the TEE.
- Connection requests are delivered through an *MQTT broker* to facilitate communications between multiple devices and servers, and should specify the UUID of the trusted application that is addressed by the requester, the URL the trusted application should connect to, and an optional port to make this connection to.

When receiving a connection request, the trusted application is solely responsible for opening the connection, so that certain (application-specific) security conditions can be verified before connecting. For example, if the trusted application decides that the timing of the request is suspicious, or if the parameters of the request do not align with the expected values, the TA can refuse this connection attempt. An overview of the message flow is shown in Figure 13.

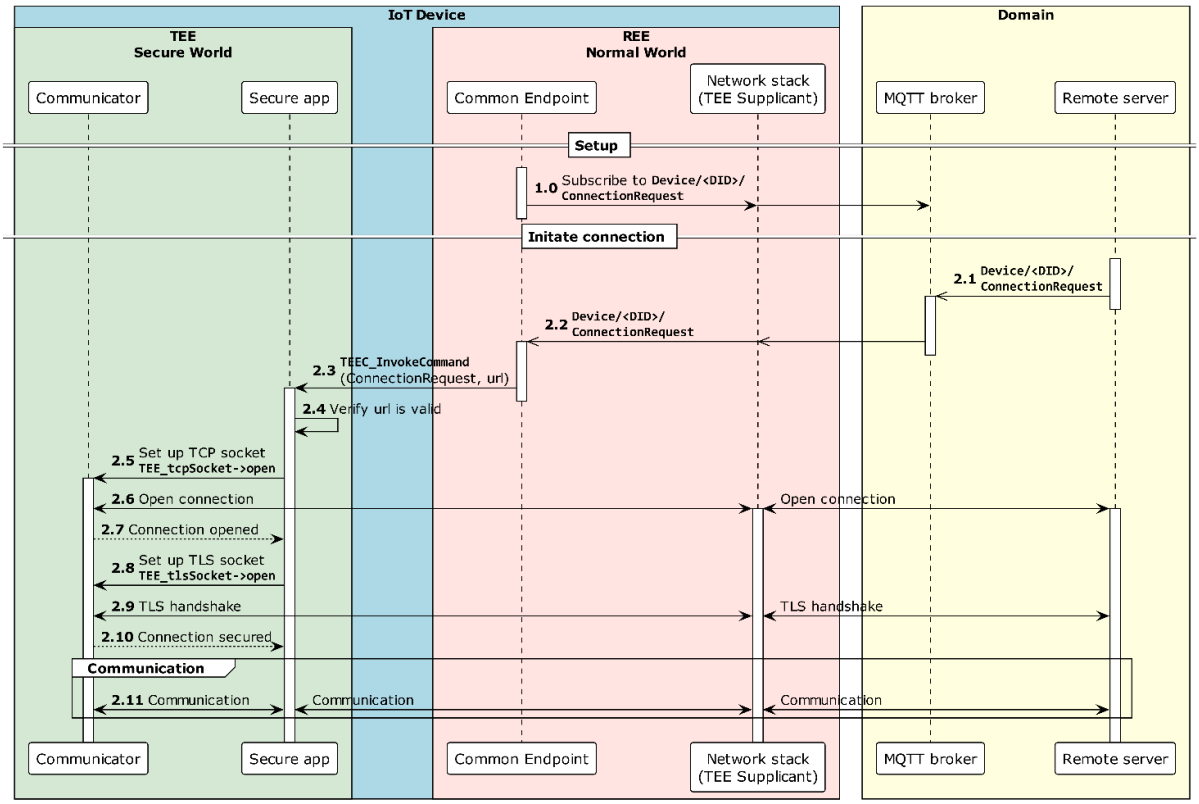

Figure 13: Message flow for a connection request to an IoT device

5.3 Use Cases

The next sections describe how the enhanced TEE can be used to more easily extend the ERATOSTHENES architecture with two applications, remote attestation and proof of secure erasure, which are classically difficult to design and implement well.

5.3.1 Remote Attestation

Remote Attestation (RA) is a technique for remotely verifying the correct operation of a device, often used with IoT devices. However, the difficulty of implementing many RA schemes on IoT devices is often immense; not only is the hardware limited due to the low-cost nature of the IoT, executing RA limits the available time and energy for tasks that are more important to the functionality of the IoT devices.

Most of the complexity of RA comes from the assumed threat model: malicious software has the same access and execution privileges as the RA software. Consequently, a multitude of complex techniques need to be employed to prevent the malicious software from interfering with the RA process, heavily influencing the total performance of the IoT device.

By moving the RA process to the TEE, the confidentiality and integrity of the TEE can instantly be leveraged to safeguard RA execution. This decreases the complexity of both the RA process and the backend services that support it, which only need to trust the TEE to trust the result of the process. Additionally, because the TEE has full access to and control over the device, more complex attestation can be performed with only limited impact on the overall performance of the device.

5.3.2 Proof of Secure Erasure

Proof of Secure Erasure (PoSE) is a process wherein a piece of memory is erased by the IoT device and a proof is generated for the verifying party that the memory actually was erased. PoSE can be used to ensure secret data is securely removed or malicious software is completely erased from the device.

As with Remote Attestation, the process of PoSE is difficult to complete securely on a device where malicious code has the same privileges as the PoSE process. Classic approaches to these problems have relied on the generation of random data, which is used to overwrite the secret data or malicious software, and a time limit to ensure malicious code cannot generate similar data before the completion of the process is expected. As a result, the overhead on both the network between the verifier and the IoT device and the computational effort for the verifier is non-negligible, making the system less scalable.

Similar to the Remote Attestation example, by moving the PoSE process inside the TEE, the security properties of that TEE are gained, with little to no cost. Additionally, because the environment in which the PoSE process is now running can be trusted, there is no need to employ techniques like explained in the previous paragraph, and a simple algorithm can be used to execute PoSE of which the result can inherently be trusted by the verifier.

5.4 Summary

The enhanced TEE capabilities offered by the ERATOSTHENES project are a key enabler for trust management in IoT systems. For example, correctly calculating the risk associated to IoT devices in the TMRA (Section 3) requires integrity of the data provided by the Trust Manager, which can be safeguarded by running the Trust Manager in a TEE and securing communications as described above. Furthermore, the TEE and secure communication channel enable securely deploying updates to the Trust Manager itself (Section 4). More in general, the provided TEE capabilities provide the necessary building blocks for confidentiality and integrity for multiple ERATOSTHENES components, namely the Trust Agent, SSI agent, and Data Protector.

6 Device Network Enrolment

In the context of the ERATOSTHENES project, the device enrolment process enables devices to obtain identities with limited lifetimes based on various technologies such as PUF, TEE, and DIDs. To enroll in a domain, a device must first produce encryption and authentication material using its Root of Trust (PUF or TEE). This material includes identity proofs such as Verifiable Credentials (VCs) and DID ownership claims, as well as owner information. The domain can check through this identity-related information trust values through the DLT deployed in the domain and the multi-domain DLT. If there is prior information related to the device from any domain, it can be used for the enrolment process and the initial trust computations. Another important aspect of device enrolment is that it binds the device's owner identity with the device's identity for accountability and nonrepudiation of actions.

Device network enrolment plays a crucial role in recovering an IoT device's identity and software components in case a device is compromised during an attack or fails. That is because the IoT device's identity is associated with a certain configuration and a user during enrolment. Recovery examines the list of applications and software installed in the enrolled device and attempts to recover the IoT device to its most genuine status possible. There are two types of enrolment: domain enrolment and cross-domain enrolment.

6.1 Components Involved in the Device Network Enrolment Process

To facilitate the device network enrolment process, a set of components from the ERATOSTHENES architecture must collaborate. These components reside on the device side and the ERATOSTHENES domain side. The following entities from the infrastructure's side participate in the enrolment process:

- The *Trust Manager and Broker (TMB)* is responsible for evaluating the trustworthiness of IoT devices and acts as an MQTT broker to enable the communication between its various submodules: the TMRA module, MUD Management module, CTI Sharing Agent, Monitoring IDS, and the Trust Manager. It is the entry point for device enrolment and continuous usage of the ERATOSTHENES trust framework.
- The *Self-Sovereign Identity (SSI) Management* module is a crucial component for the enrolment of devices, issuing the VCs used by devices to show their identity attributes during the authentication and authorization processes.
- The *SC/DLT* enables different components and features in the ERATOSTHENES architecture to share and verify public information needed for their proper functioning, such as registering and retrieving DIDs, CTI data, Trust Scores, and Identity verification-related information. Therefore, it makes the process of sharing information secure, reliable, verifiable, and transparent.
- The *Management and Recovery* module manages devices in the ERATOSTHENES ecosystem, particularly regarding monitoring and recovery processes. During enrolment, it participates in the linkage of the device's identity to its introducer or owner.
- The *Device introducer/owner* though not per se a component of the architecture, plays a key role in the device's enrolment process. Specifically, the introducer is a human user who associates themselves with the device they install in the IoT ecosystem for accountability. They are the IoT ecosystem's security manager.
- The *Inter-DLT* is a platform that enables the interactions between DLTs from different ERATOSTHENES domains.

The following entities from the device's side participate in the enrolment process:

- The *Data Protector (ADP)* supports other components or subcomponents with features for handling data encryption that must be securely stored employing device hardware or software security features (e.g., TEE). Moreover, the data protector provides features of secure data (e.g., private key) exchange for fine-grained, confidential information sharing.
- The *Trusted Execution Environment (TEE)* (Section 5) is an area on a device's main processor separated from the system's main Operating System, creating a secure world for executing trustworthy code. It ensures that data is stored, processed, and protected in a secure and isolated environment. It can generate cryptographic material (e.g., keys and credentials) that serves as a basis for authentication in ERATOSTHENES. It cooperates with the Data Protector to enable other components to use said credentials.
- The *PUF client* handles the interconnection/interaction with the PUF Auth Server securely, acting as a root of trust for the identification/authentication of the device. It can generate cryptographic material that serves as a basis for authentication in ERATOSTHENES, cooperating with the Data Protector to enable the usage of said credentials by other components (e.g., SSI Agent).
- The *SSI Agent* is responsible for supporting the flows for collecting and manipulating identity data, including cryptographic operations like the generation of Zero-Knowledge Proofs (ZKP). It will manipulate VCs and generate Verifiable Presentations (VPs) for a given access policy following a Self-Sovereign Identity approach using Decentralized Identifiers (DID) and VCs, both published by the World Wide Web Consortium (W3C). Additionally, it will take advantage of the TEE and the Data Protector module for the secure storage and retrieval of the necessary private cryptographic material.
- The *Trust Agent* handles multiple activities related to Trust Management on the device's side, such as software boot and monitoring. Mainly, it handles interactions with the TMB to calculate trust and reputation during interactions, including the initial enrolment of the device in the system that results in the initial trust score evaluation and the culmination of the enrolment process.

6.2 IoT Device Network Enrolment Background

Enrolling an IoT device is a key process within its lifecycle, serving as an enabler for all functionalities that will occur during the operational time. Before enrolment occurs, it is assumed that the initial bootstrapping of devices during the manufacturing process is carried out, and right after the device is deployed, the initial bootstrapping is finished. The ERATOSTHENES enrolment covers three cases depending on the device root of trust (PUF, TEE, or none), demonstrating the flexibility of the ERATOSTHENES architecture to deal with heterogeneous devices and network settings.

To achieve the above objectives, the enrolment process deals with identity and trustworthiness. Regarding identity, the ERATOSTHENES enrolment process follows the SSI paradigm, where each device manages its own credentials that comprise its identity in the IoT ecosystem. This facilitates fine-grained attribute-based control when devices access services in the IoT ecosystem. To obtain the aforementioned credentials, the device has to undergo an onboarding process, where it registers its DID document, which contains the public keys that enable the device's identification and the VCs that contain its identity attributes through an issuance process against the SSI Management component in the domain's infrastructure. During this process, the device is subjected to a proof of identity process, where the root of trust for identifying the device plays a key role. Ideally, PUF-based authentication is used, but alternatives are possible and are considered. Consequently, in this process, the device's PUF authentication module facilitates device authentication and identification, while the SSI agent deployed on the device manages

the interactions and processes relevant to the DIDs and VCs, supported by the Data Protector as a secure environment for storing and managing cryptographic material. On the ecosystem's infrastructure side, the SSI management module plays the role of the credential issuer, providing devices with the identity material as a result of the enrolment. A DLT supports the enrolment as a verifiable data registry, while manufacturer services such as PUF authentication servers facilitate the initial identity proof.

The device's enrolment process includes an additional step for the inclusion of the device in the ecosystem's trust framework. The device, through its Trust Agent, will connect to the domain's TMB and be given an initial evaluation of its trustworthiness that depends on parameters including but not limited to its identity, the endorsement of the introducer (e.g., a security domain manager), or threat modeling and risk assessment (with the TMB's subcomponents collaborating). As indicated before, a crucial source for this evaluation is the trustworthiness of the identification mechanism of the device through the identity's root of trust. Once the evaluation process is completed, an initial trust score is assigned to the device. It will be updated throughout its operation in the domain depending on events related to its behavior or the discovery of new threats.

Only when both processes are completed, and a device has the corresponding identity and trust information, is it fully enrolled and can interact with other elements in the domain as part of its operational phase.

6.3 IoT Devices Single Domain Network Enrolment Operation Flow

The IoT Devices Network Enrollment mechanism has three phases of operations: *Phase 1 – Onboarding and Creation of Public DID*, *Phase 2 – Introducer-device pairing*, and *Phase 3 – Device enrolment and initial calculation of trust*. Enrolling IoT devices within the ERATOSTHENES framework is accompanied by the completion of formal registration as a recognized ERATOSTHENES ecosystem member.

During the first phase, two suboperations occur: generating a public DID and registering it in the SSI Management. Creating a public DID involves generating a unique DID for the device and crafting a DID document that complies with the specifications outlined in the DID W3C standard. In this sub-process, the SSI Agent (ID client) component must utilize the associated cryptographic keys within the system to generate the DID Document. The cryptographic material is generated, stored, and obtained through three distinct sources: Hyperledger Aries (legacy), TEE, and PUF cryptographic material generation. The source can be selected by configuring an environmental variable.

The value of this variable is established based on the specific requirements and needs of the ERATOSTHENES ecosystem. This flexibility in configuration allows the SSI Agent component to adapt to the unique requirements of the ERATOSTHENES ecosystem.

When the variable is set to "ADP + PUF", the SSI Agent utilizes the PUF client library through the ADP component to acquire physically unclonable cryptographic material. However, when the value is set to "ADP + TEE", the SSI Agent utilizes the TEE through the ADP. Conversely, when the value is set to 'default', the SSI Agent component generates its own cryptographic material by utilizing the SSI standard by the W3C[9].

Once the SSI Agent possesses the cryptographic keys, it generates the DID Document. However, before generating the DID Document, the DID itself is created, offering two distinct methods for publishing the public DID:

- `did:web:` Adhering to the did:web standard.

[9] `https://www.w3.org/TR/did-core/`

- `did:erat:` A customized method developed for the ERATOSTHENES project that enables the publication of the DID into a permissioned blockchain implemented with Hyperledger Fabric.

Upon successfully publishing a new DID linked to its identity, the next step involves registering the IoT device in the SSI Management module. Afterward, the device generates a VC that includes its unique footprint. It then requests the SSI Management module to issue the credential.

The second phase is the Introducer-Device pairing, during which the device's certificate generated during Phase 1 is associated with the introducer for accountability purposes. A typical envisioned association between the introducer and the device occurs as follows. Initially, the device requests from the Management and Recovery service to be linked to the introducer. Upon receiving that request, the Management and Recovery service requests the Introducer to authenticate. The introducer authenticates to the Management and Recovery Service and then receives a request to approve the device's DID from the latter. The introducer approves the device, resulting in a signed DID, which the Management and Recovery service sends to the device.

Finally, in the third phase, the Device Enrolment and Initial Calculation of Trust commences, during which the device sends its trust data along with its, signed by the introducer, DID, to the TMB. The TMB calculates the device's initial trust score and, via its gRPC client, sends it along with the device's DID to the gRPC server of the DLT. The gRPC server invokes the chaincode that writes the trust score to the DLT. Once the trust score is recorded in the DLT, the chaincode will send an acknowledgment to the TMB. Afterward, the TMB forwards the trust score to the IoT device.

6.4 IoT Devices Cross-Domain Network Enrolment Operation Flow

The cross-domain network enrolment focuses on the enrolment of a device from domain A to domain B. The process starts with the SSI Agent generating a VP from a VC obtained from domain A. Then, the information related to the enrolment of the device to domain A, along with the VP, is sent to the SSI Management module of domain B, which in turn Validates the identity proofs and retrieves the issuer's DID from the inter-DLT service. Once the SSI management retrieves the DID of the issuer from domain A, the respective VP is validated. Upon successful validation, the SSI Management sends the Device Enrolment Response to the SSI Agent. Afterward, domain B's Management & Recovery service links the device's identity with its owner's. Once the linking is done, the Trust Agent sends the trust-related data to the TMB, which checks in the Inter DLT for prior trust evaluation-related data and retrieves them through smart contracts. Then, it computes the device's initial trust score for domain B by considering its context and the trust data from domain A. The calculated trust score is stored on the DLT of domain B, and then the TMB sends the enrollment result to the device.

6.5 Summary

In summary, enrolment is pivotal in enhancing accountability within IoT ecosystems, fostering traceability, and reinforcing security auditing procedures. This is achieved by linking the device owner's identity to their IoT device, ensuring accountability and non-repudiation of actions. Additionally, all outcomes stemming from the three phases of the enrolment process are logged onto the DLT, serving as an immutable ledger. Thus, regardless of how much malicious actors endeavor to disassociate themselves from the connected IoT device, the process remains resilient, making it arduous to complete disassociation. Furthermore, the enrolment process provides valuable insight into a device's trustworthiness once it joins the IoT ecosystem, enabling prospective IoT service consumers to make informed decisions when choosing an IoT service provider.

7 Conclusion

IoT-based applications involve devices that are deployed typically in uncontrolled and uncontrollable environments. For example, in a smart traffic system, IoT devices are installed in the public domain as part of the infrastructure (e.g., a traffic light) or within vehicles, in a remote patient monitoring service, devices are handed out to patients who use them at home, and in a smart manufacturing case, devices are deployed at large scale in complex factory settings.

Distributed trust—the ability to evaluate trust from an externalized perspective—therefore is the only viable approach in such environments. Yet, there is no single-size-fits-all solution to establish trust in heterogeneous devices and dynamic environments, and careful application-specific trade-offs must be made by application engineers and operators, taking into account many factors such as device capabilities (e.g. hardware provisions for trusted execution, computational cost, battery cost, bandwidth), level of security and assurance, and residual risk.

This chapter presented an architecture—and the integration of diverse architectural enablers therein—to accomplish exactly this. This solution is customizable, as this toolkit of diverse technologies can be selected and composed in a mix-and-match manner to accomplish trust in a way that is tailored to the specific use case at hand. A dynamic calculation of trust is performed by the Trust Manager and Broker (TMB), which considers these enablers in terms of their trust contributions and residual risks. This trust qualification is then used to inform run-time systems (for example to inform access control decisions), and even to drive proactive or reactive actions (e.g., dynamic re-deployment or rollback of software state on a specific device).

While the ensuing tool kit by no real means can be considered complete, it does yield a diverse and versatile set of technological enablers, which have been successfully technically integrated within the scope and intent of the ERATOSTHENES project.

Acknowledgments

This work is partially funded by Research Fund KU Leuven, and the European Union's Horizon 2020 research and innovation program under grant agreement No 101020416.

References

[1] G.-H. Tzeng and J.-J. Huang, *Multiple Attribute Decision Making: Methods and Applications*. CRC press, 2011.

[2] S. Chakraborty, "TOPSIS and Modified TOPSIS: A Comparative Analysis," *Decision Analytics Journal*, vol. 2, 2022.

[3] Z. Shi, K. Graffi, D. Starobinski, and N. Matyunin, "Threat Modeling Tools: A Taxonomy," *IEEE Security & Privacy*, vol. 20, no. 4, 2021.

[4] D. Van Landuyt, L. Sion, E. Vandeloo, and W. Joosen, "On the Applicability of Security and Privacy Threat Modeling for Blockchain Applications," in *International Workshop on SECurity and Privacy Requirements Engineering (SECPRE)*, 2019.

[5] S. Verreydt, D. Van Landuyt, and W. Joosen, "Expressive and Systematic Risk Assessments with Instance-Centric Threat Models," in *ACM/SIGAPP Symposium on Applied Computing (SAC)*, 2023.

[6] S. Verreydt, D. V. Landuyt, and W. Joosen, "Run-Time Threat Models for Systematic and Continuous Risk Assessment," *Software and Systems Modeling*, 2024.

[7] H. Song, R. Dautov, N. Ferry, A. Solberg, and F. Fleurey, "Model-Based Fleet Deployment in the IoT–Edge–Cloud Continuum," *Software and Systems Modeling*, vol. 21, no. 5, 2022.

[8] H. Song, R. Dautov, N. Ferry, A. Solberg, and F. Fleurey, "Model-Based Fleet Deployment of Edge Computing Applications," in *ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, 2020.

[9] R. Dautov and H. Song, "Context-Aware Digital Twins to Support Software Management at the Edge," in *International Conference on Research Challenges in Information Science*, 2023.

[10] R. Dautov, H. Song, and N. Ferry, "Towards a Sustainable IoT with Last-Mile Software Deployment," in *IEEE Symposium on Computers and Communications (ISCC)*, 2021.

[11] R. Dautov and E. J. Husom, "Raft Protocol for Fault Tolerance and Self-Recovery in Federated Learning," in *International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, 2024.

[12] T. Van Eyck, H. Trimech, S. Michiels, D. Hughes, M. Salehi, H. Janjuaa, and T.-L. Ta, "Mr-TEE: Practical Trusted Execution of Mixed-Criticality Code," in *International Middleware Conference: Industrial Track*, 2023.