PRIVATEER

Privacy-first Security Enablers for 6G Networks

Deliverable 5.1

Distributed attestation, identity and threat sharing enablers – Rel. A.

DRAFT – Pending approval by the Smart Networks and Services Joint Undertaking (SNS JU)

Co-funded by the European Union

6G SNS

PRIVATEER has received funding from the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme under Grant Agreement No. 101096110

Space Hellas SA, NCSR "Demokritos, Telefónica I&D, RHEA System SA, INESC TEC, Infili Technologies PC, Ubitech Ltd, Universidad Complutense de Madrid, Institute of Communication and Computer Systems, Forsvarets Forksninginstitutt, Iquadrat Informatica SL, Instituto Politecnico do Porto, ERTICO ITS Europe

PRIVATEER

Deliverable 5.1

Distributed attestation, identity and threat sharing enablers – Rel. A.

Deliverable Type
Report

Month and Date of Delivery
May 20th 2024

Work Package
5

Leader
UBITECH

Dissemination Level
Public

Authors
Anna Angelogianni, Nikos Fotos, Thanassis Giannetsos, Manos Kalotychos, Stefanos Vasileiadis (UBI)

Programme	Contract Number	Duration	Starting Date
Horizon Europe	101096110	36 months	January 2023

Contact Us
privateer-contact@spacemaillist.eu

Co-funded by the European Union
GGSNS

PRIVATEER has received funding from the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme under Grant Agreement No. 101096110

Contributors

Name	Organization
Stella Dimopoulou	NCSRD
Pedro Sousa, António Pinto, Pedro Pinto	INESC TEC
Ilias Papalamprou, Dimitrios Danopoulos, Dimosthenis Masouros, Dimitrios Soudris	ICCS
Anna Angelogianni, Nikos Fotos, Thanassis Giannetsos, Manos Kalotychos, Stefanos Vasileiadis	UBITECH

Reviewers

Name	Organization
Ilias Papalamprou, Dimitrios Danopoulos, Aimilios Leftheriotis, Dimosthenis Masouros	ICCS
Pedro Sousa, António Pinto	INESC TEC

Copyright and Disclaimer

Version History

Version	Date	Modifications
1.0	20/05/2024	Final version

List of Acronyms

Acronum	Description
AA	Attestation Agent
AK	Attestation Key
ALTO	Application-Layer Traffic Optimization
ACA-Py	Aries Cloud Agent Python
B5G	Beyond 5G
BFT	Byzantine fault tolerance
CA	Consortium Agreement
CIPs	Cloud Infrastructure Providers
CIV	Configuration Integrity Verification
CTI	Cyber Threat Intelligence
DID	Decentralized Identifier
DLT	Distributed Ledger Technology
DSP	Digital Signal Processing
eSE	embedded Secure Element
EVM	Ethereum Virtual Machine
FPGA	Field Programmable Gate Array
GA	Grant Agreement
IdM	Identity Management
IoT	Internet of Things
KRPE	Key Restriction Usage Policy Engine
KRUP	Key Restriction Usage Policies
LoA	Level of Assurance
LoT	Level of Trust
MNO	Mobile Network Operator
NEF	Network Exposure Functions
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PL	Programmable Logic
PoA	Proof of Authority
PoT	Proof of Transit
PoW	as Proof of Work
PUF	Physical Unclonable Function
RoT	Root of Trust
RR	Revocation Registry
RTM	RoT for Measurement
RTR	RoT for Reporting
RTS	RoT for Storage
SCs	Smart Contracts
SCB	Secure Context Broker

SGX	Software Guard Extensions
SSI	Self-Sovereign Identity
SLA	Service Level Agreement
SGX	Software Guard Extensions
TC	Town Crier
TCB	Trusted Computing Base
TEE	Trusted Execution Environment
TSP	Telecommunication Service Providers
TTP	Trusted Third Party
URI	Uniform Resource Identifier
VC	Verifiable Credential
VFs	Virtual Function(s)
VNFs	Virtual Network Function(s)
VP	Verifiable Presentation
VPE	Verifiable Policy Enforcer
W3C	World Wide Web Consortium
ZKP	Zero Knowledge Proof

Executive Summary

Beyond 5G (B5G) technologies are currently undergoing substantial transformations as new functionalities emerge and new challenges arise. As we move towards the implementation of 6G technology, it becomes evident that strong security measures are necessary. A particularly interesting concept is Zero Trust. This concept operates under the assumption that no entity can be inherently trusted, and therefore all entities are required to provide evidence of their trustworthiness (i.e., correct functioning). By adopting the concept of Zero Trust, Mobile Network Operators (MNOs) may mitigate the risks posed by both external and internal threats, thus improving their overall cybersecurity posture. Zero Trust is crucial for defending against the dynamic threat landscape as well as the diverse systems and environments envisaged in B5G use cases. To adhere to this notion, a strict approach to security is crucial. Hence, apart from the more traditional threat intelligence approaches, PRIVATEER is further focusing on the development of attestation mechanisms across the network continuum.

To promptly identify and address any deviations from the expected configuration, MNOs, Cloud Infrastructure Providers (CIPs), or Telecommunication Service Providers (TSPs) can implement strong attestation mechanisms, operating both at a remote and a local level. This enables the mitigation of risks related to unauthorized modifications that could result in data breaches or service disruptions. To achieve greater protection, attestation mechanisms should extend to further support runtime configuration integrity verification, apart from the traditional secure bootup. Towards this direction, PRIVATEER designs include the attestation of the correct configuration of edge accelerators (i.e., FPGAs) as well as the enabling of the ongoing monitoring and evaluation of the integrity of both the virtualised services as well as the entire containers throughout their entire operational lifecycles. The aforementioned designs further support different Levels of Assurance (LoAs) in accordance with existing standards. These levels define the verification levels achieved for each service or infrastructure component.

Identity verification, authentication, and authorization are also essential for allowing only entities with the appropriate attributes to access a service or information related to the level of trustworthiness evidence. Self-Sovereign Identity (SSI) represents a progression from user-centric identity, allowing users to have complete control over their own identity without the need for a central authority. Towards this direction, blockchain technologies are leveraged in PRIVATEER both in regard to identity management as well as trustworthy data exchange as it pertains to trustworthiness evidence, leveraged for Trust Assessment. Blockchain by design offers a transparent and auditable means for information exchange, apart from confidentiality and integrity protection. one of Zero Trust

Furthermore, the EU's 6G vision for the next generation of telecommunications places great importance on privacy, implementing the necessary measures to provide advanced security while upholding privacy. To adhere to the privacy requirements, PRIVATEER adopts Zero Knowledge Proofs-based schemes, which provide a verifiable means to assert the configuration integrity of a virtualised service without revealing any details regarding the exact evidence that was obtained. Additionally, a Trust Exposure Layer is proposed to harmonise the acquired information, giving access to external (to the infrastructure) parties strictly to information regarding the level of trust. Finally, Searchable Encryption mechanisms are further leveraged in Cyber Threat Intelligence search operations, for protecting critical infrastructures by conducting searches over encrypted data.

Table of Contents

List of Figures

List of Tables

1 Introduction

In the rapidly evolving landscape of B5G technologies, novel functionalities emerge along with new challenges. While protocol designs emphasize security, the dynamic nature of cyber threats demands continuous adaptation and enhancement of the existing protective measures. As we transition to 6G, the imperative for robust security measures becomes even more evident. These measures refer not only to the offered services but also to the infrastructure where these services are deployed. Modern systems are envisioned to adhere to the **Zero Trust notion**. This notion is based on the assumption that no entity can be inherently trusted; hence, all entities must provide proofs over their trustworthiness and correct operation. By adopting Zero Trust principles, organizations can minimize the risks caused by insider threats and enhance overall cybersecurity posture.

In the context of B5G systems, Zero Trust becomes instrumental in protecting against evolving cyber threats and safeguarding critical assets across distributed and heterogeneous environments. However, achieving Zero Trust-based security requires a multifaceted approach. Ensuring the **integrity and resilience** of both the services as well as the underlying infrastructure is paramount for safeguarding against sophisticated cyber threats that exploit vulnerabilities in interconnected services or even interconnected networks. As such, a comprehensive approach to security is essential, encompassing both **remote** and **local attestation** to **verify the configuration integrity of virtualized services**, **Virtual Functions (VFs), Virtual Network Functions (VNFs), and the underlying heterogeneous infrastructure spanning from the core site to the edge**. This integrity verification process should **extend beyond secure bootup, covering runtime configuration** as well, which allows continuous monitoring and assessment of system and network integrity **throughout their operational lifecycles**. Aligning with established standards such as those articulated by ETSI, the establishment of different Levels of Attestation (LoAs) becomes crucial, defining the accomplished verification levels for each service or infrastructure component.

By implementing robust attestation mechanisms at both remote and local levels, Mobile Network Operators (MNOs), Cloud Infrastructure Providers (CIPs), or Telecommunication Service Providers (TSPs) can promptly identify and address any deviations from the expected configuration. This proactive approach mitigates the risk of unauthorized modifications which could lead to data breaches, or service disruptions, thereby enhancing the overall security posture of the network infrastructure.

Furthermore, identity verification, authentication and authorization is another crucial aspect in this regard, ensuring that **only actors** (i.e., including both users and/or services) **possessing the correct attributes can gain access to a service or information regarding the level of trust or the trustworthiness evidence**. Self-sovereign identity

(SSI) is the next step beyond user-centric identity according to the European Commission [1]. Compared to centralized identity verification and authentication schemes, SSI enables the user's sovereignty and control over its identity, creating user autonomy, without the need of a central party that could be a single point of failure, while providing the ability to use an identity across multiple locations. Towards this end, the exploration of Decentralized IDentities (DID) by PRIVATEER, as specified by the W3C, presents a modern, decentralized, and lightweight alternative to established authentication methods. This innovative approach, encompassing DIDs, verifiable credentials (VCs), and verifiable presentations (VPs), extends beyond individuals to institutions and devices within the Internet of Things (IoT) ecosystem, further enhancing the security and trustworthiness of digital interactions.

Simultaneously, privacy stands as a significant pillar of the EU's 6G vision for the next generation of telecommunications. Fostering security measures that protect all actors involved, including users, services, or even the entirety of the underlying infrastructure, while preserving privacy, is pivotal. This focus on privacy underscores the essence of preserving individual autonomy and data confidentiality in the evolving digital landscape. Within this multifaceted ecosystem, privacy assumes a pivotal role, extending its relevance beyond end-users to encompass all engaged stakeholders. By embedding privacy as a fundamental principle, 6G endeavours to foster trust, transparency, and accountability, thereby ensuring the ethical deployment of future telecommunications technologies. Towards the direction of safeguarding privacy, Zero Knowledge Proofs enable verification of the configuration integrity of Virtual Functions (VFs) or Virtual Network Functions (VNFs) without the need to disclose specific details of the extracted evidence. By leveraging Zero Knowledge Proofs, organizations can authenticate the integrity of system components while preserving sensitive information, thereby enhancing privacy and confidentiality in the verification process.

1.1 Document structure

The rest of the document is structured as follows: Chapter 2 provides the architectural details over the WP5 as it pertains to its components, the internal structure and communication flows, along with the functional specifications. Chapters 3, 4, 5, 6 and 7 detail on the runtime attestation for virtualised services and containers, attestation in edge accelerators, blockchain for exchange of trustworthiness evidence, distributed identity management and privacy-preserving CTI sharing components respectively. Lastly, Chapter 8 draws the conclusions.

[1] https://joinup.ec.europa.eu/collection/ssi-eidas-bridge

2 Security & identity management components within PRIVATEER architecture

2.1 Overview of the PRIVATEER architecture

An overview of the consolidated PRIVATEER architecture as it pertains to the distributed attestation, identity and threat sharing enablers is illustrated in Figure 1. The overall PRIVATEER architecture follows the notions of the HEXA-X flagship project [1] definitions, offering though security and privacy extensions for dynamic and evidence-based trust assessment. It shall be noted that the proposed framework is designed to remain agnostic to the specific orchestrator technology employed, ensuring flexibility and compatibility across various infrastructures and environments.

2.1.1 PRIVATEER Layers and Functionalities

The depicted architecture is separated into different layers.

The **infrastructure/asset layer** is the fundamental level that consists of the physical equipment and assets necessary for the system to function (i.e., infrastructure elements). This layer comprises a wide array of devices, such as IoT devices, servers, switches, and routers, among others. The servers offer the hosting environment for the virtualized microservices and containerized applications that form the core of the PRIVATEER ecosystem. Among this hardware, Field Programmable Gate Arrays (FPGAs) are included for efficiently supporting applications such as Digital Signal Processing (DSP), Deep Learning, etc. Furthermore, these servers may incorporate hardware-enabled Trusted Execution Environments (TEEs), such as Intel SGX, to establish secure and isolated execution environments known as enclaves. TEEs ensure that critical workloads operate within a restricted and isolated space, inaccessible to other software running on the same host.

This capability significantly enhances the system's security by safeguarding against unauthorized access and ensuring the integrity of essential processes and data. Essentially, the hardware-enabled TEEs provide the Root of Trust (RoT) capabilities within PRIVATEER, facilitating the secure execution of critical binaries. In PRIVATEER, Gramine[2] is leveraged, as the trust anchor, for instantiating all newly developed secure life-cycle management controls, within secure enclaves, in a lightweight manner. The Gramine technology was chosen because it has the ability to convert a software binary

[2] https://gramineproject.io/

into a trusted equivalent that runs in a separate environment. This conversion can be done without making any updates to the code and without impacting its dependencies on other parts of the software stack. In essence, it enables the execution of any binary file within a secure and isolated environment known as an enclave. This technology is a recent development that falls under the established SGX TEE technology. In conjunction with Gramine, enclave-cc is utilised for the launching of confidential containers.

Examining the Intel SGX-enabled servers, which provide the hardware-enabled TEE, there is a clear distinction between processes executed in the "trusted" and "untrusted" domain within the **Security Probe**. The "trusted" part of the Security Probe incorporates the attestation-related tasks as well as the validation of the active key restriction usage policies. It shall be noted that the "trusted" part further provides the **Secure APIs**, exposing an interface to the untrusted world. This interface will serve as a focal point for future research efforts focused on exporting attestation results, including attestation evidence for specific trust properties (e.g., integrity), directly from the trusted world. As mentioned in the D2.2 [2], the Attestation Agent residing with the Security Probe (i.e., the Prover) conforms to a zero-knowledge based scheme, ensuring the configuration state of the microservice by exporting a proof of correctness that includes, the key restriction usage policies instead of the traces. This proof of correctness is received by the Secure Oracle that acts as the Verifier. The protocol is challenge-based, with the Verifier initiating the process with a nonce. This approach avoids the disclosure of unnecessary information, adhering to the privacy-preserving notions of PRIVATEER.

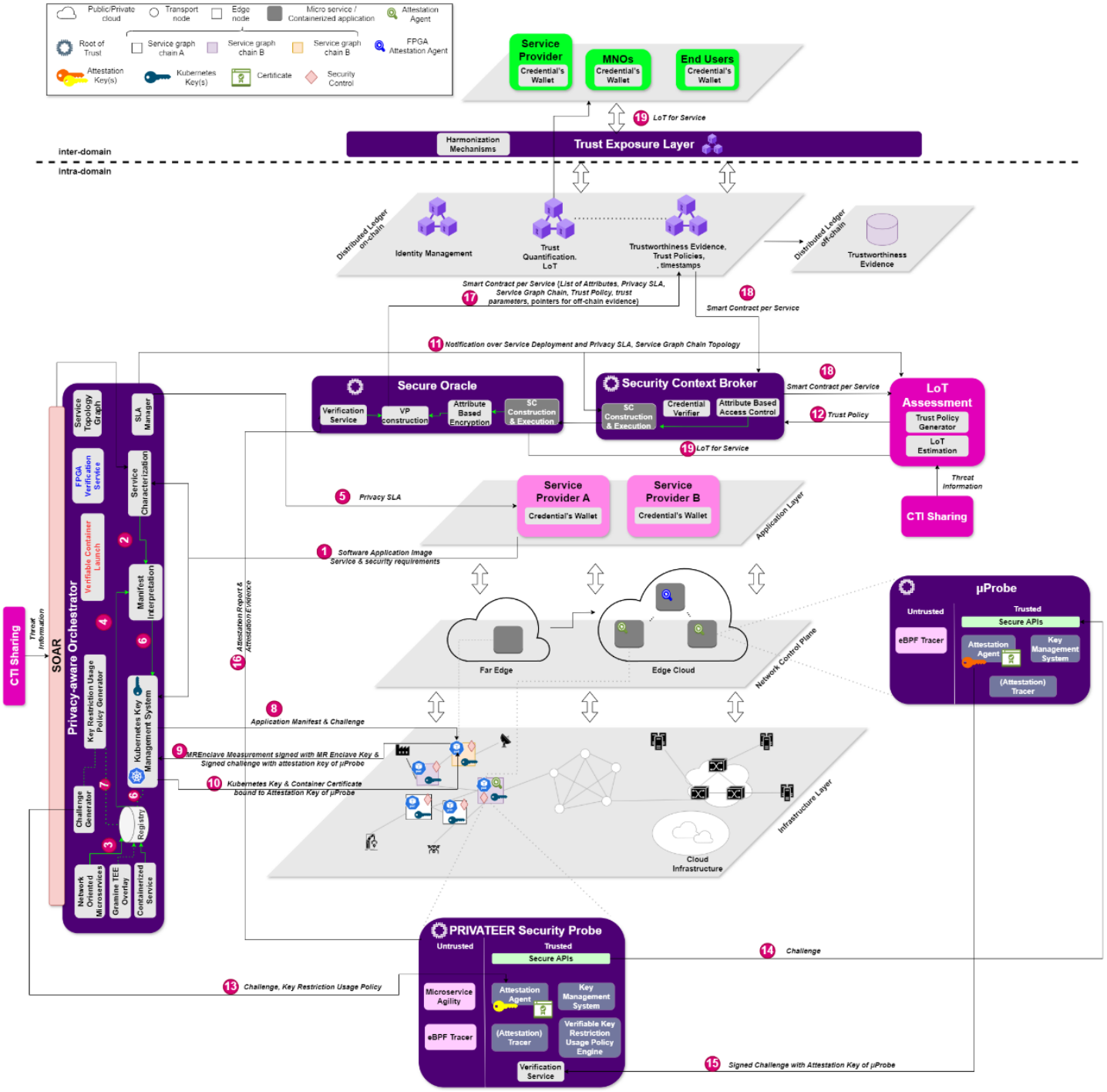

Figure 1 - PRIVATEER's Distributed attestation, identity and threat sharing enablers Architectural Overview

On the other hand, the "untrusted" part of the Security Probe encompasses the **eBPF tracer**[3] which collects the configuration integrity traces during runtime (i.e., from routines and services), and the **Microservice Agility** which is communicating with the Orchestrator and the Secure Oracle. This communication is expected to facilitate

[3] https://ebpf.foundation/

secure updates from the Orchestrator to the Security Probe, without requiring the latter to be re-enrolled. This feature, planned for integration in the platform's Release B, enables seamless updates such as the inclusion of updated versions of the Attestation Agent or the introduction of new components in the Security Probe. It shall be noted that apart from the Security Probe, attestation services are also supported by the **FPGA devices**, to ensure authenticity and integrity of the binary.

Moving to the **network control plane layer**, this is where the deployment of virtualized microservices and containerized applications occurs. Servers designated for hosting microservices may be positioned either at the edge or the cloud side of the network, depending on specific application requirements and the desired proximity to end-users or data sources. PRIVATEER employs tailored attestation mechanisms designed explicitly to verify the integrity and authenticity of these services, comparing their runtime measurements to predefined reference values. This process ensures that the deployed services operate as anticipated, thereby strengthening trust in the system among users and stakeholders. Hence, in addition to the monitoring and attestation functions provided by the Security Probe for both the container and the underlying infrastructure, **µProbes** are deployed within the containers to ensure the configuration integrity of the containerized application. These µProbes are equipped with their own set of components, including eBPF tracers in the untrusted world, as well as Attestation Tracer, Attestation Agent, Key Management System, and Secure APIs in the trusted part. As a result, they independently collect configuration integrity traces, sign them with their respective attestation keys, and transmit them to the Security Probe for verification. This setup implies that the Security Probe not only reports its own attestation results but also incorporates the attestation results from the µProbes. More information regarding the Secure Launching of a container is elaborated in Section 2.1.2.1 of the present deliverable.

It shall be clarified that the Security Probe and the µProbe serve distinct, yet complementary roles within the PRIVATEER platform's security architecture. The first, the Security Probe, operates at the container level, encompassing components like the Microservice Agility, the Attestation Agent, the Key Restriction Usage Policy Engine and the Verification Service. Its primary function is to ensure the integrity and security of the containerized environment, facilitating runtime attestation of the entire container and enforcing security policies. The µProbe, on the other hand, is tailored for individual microservices executed within the containers, providing a finer-grained approach to security monitoring and attestation. The combination of the two components provides a holistic security framework, safeguarding both the containerized environment and microservices against threats and unauthorized modifications.

The secure and seamless deployment process of the microservices is supported by the **Orchestrator**. The latter receives service and security requirements from the Service Provider (SP). Additionally, the Orchestrator receives Cyber Threat Intelligence (CTI)

information through the Security Orchestration, Automation and Response (SOAR) element and can thus apply Security controls. It then translates these high-level requirements into concrete security characterizations, determining the specific security controls to be applied. For instance, it may decide to deploy the service with a Security Probe installed to offer runtime attestation. This process results in the creation of an Interpretable Manifest and Privacy Service Level Agreements (SLAs), which outline the security measures and performance expectations for the service based on both its characteristics and the underlying network and infrastructure. The Orchestrator's decision-making is driven by the goal of optimizing resource allocation to ensure that the necessary security enablers are provided as required for the execution of the service, thereby meeting performance and security objectives.

With direct access to all resources within the infrastructure, the Orchestrator will be empowered in the Release B of the PRIVATEER platform to perform updates related to the Security Probe. This includes adding or removing internal modules within the Security Probe and updating their versions, such as the Attestation Agent. It is important to note that in the Release A of the PRIVATEER platform, updates regarding the Security Probe are not executed during runtime. Instead, the container needs to be relaunched to introduce an updated Security Probe.

Furthermore, the Orchestrator initiates the construction of a smart contract that is published to the ledger via the Security Context Broker (SCB) and the Secure Oracle, ensuring transparency and immutability. This smart contract includes the agreed Privacy SLA between the Orchestrator and the Service Provider, as well as information related to network awareness (i.e., service graph chain). Additionally, the Orchestrator notifies the LoT assessment component upon the successful deployment of a new service, initiating the process of creating Trust Policies and uploading them to the ledger.

The **Level of Trust (LoT) Assessment** is the component responsible for defining the Trust Policies for a specific service graph chain and performing the trust evaluation. More specifically, each service may require a different set of parameters, from specific trust sources. Therefore, it is imperative to define a Trust Policy based on the LoT Assessment, which will subsequently be made available on the ledger, through the Security Context Broker (SCB) and the Security Probe.

The **Security Context Broker (SCB)** facilitates the construction of smart contracts for the Secure Oracle, providing the list of attributes, the Privacy SLA and the service graph chain topology as received from the Orchestrator, as well as the Trust Policies as received from the LoT Assessment component. It further facilitates the querying of the ledger on behalf of the LoT Assessment component and the Orchestrator.

It shall be noted that if the service graph chain is modified then a new Trust Policy and smart contract shall be constructed by the Orchestrator and the Security Context

Broker through the Secure Oracle respectively. For auditability purposes the new smart contract will include a pointer to the old smart contract.

The **Secure Oracle** operates with the support of a Trusted Execution Environment (TEE); thus, acts as an integral part of the PRIVATEER Trusted Computing Base (TCB). As such, it is inherently considered a trusted component within the system. The Secure Oracle performs multiple tasks. Firstly, it receives and validates attestation reports generated by the Security Probes, ensuring the authenticity of the information provided. Additionally, the Secure Oracle is responsible for generating smart contracts, which contain comprehensive information required for trust assessment processes. These smart contracts, dedicated to specific services, include essential information such as i) Privacy Service Level Agreements (SLAs), ii) the service graph chain, iii) trust policies, iv) trust parameters (including signed attestation reports), v) access control attributes, and vi) pointers to off-chain storage for failed attestation evidence. To uphold the integrity of these smart contracts, they are hashed and signed with the Secure Oracle's private key, allowing any entity accessing the Ledger to verify their authenticity using the Secure Oracle's public key. Whenever the LoT Assessment accesses information stored in the Distributed Ledger Technology (DLT), through the SCB, for conducting a new trust evaluation, the outcome of this assessment can be incorporated into the smart contract. This integration occurs through communication between the LoT Assessment component and the Secure Oracle, allowing for the seamless inclusion of the evaluation result within the smart contract. The LoT and the Orchestrator may access the DLT through the Security Context Broker.

It shall be noted that PRIVATEER further leverages **Distributed Ledger Technology (DLT)** for identity management. More specifically the concept of Self-Sovereign Identity (SSI) through Decentralized Identifiers (DIDs) is adopted in PRIVATEER, offering to individuals' ownership and control over their digital identities. DIDs provide a decentralized, tamper-proof method for verifying individuals' identities. These identifiers are stored on the Ledger, a distributed database that maintains a transparent and immutable record of all transactions and identity-related activities. Moreover, PRIVATEER incorporates the use of Verifiable Credentials (VCs), which are digitally signed documents that attest to the authenticity of specific identity attributes or claims. VCs enable individuals to prove their identity or qualifications to third parties without disclosing unnecessary personal information. The issued VCs are securely stored within the trusted boundaries of a secure wallet, which may be hosted by a trusted Service Provider or a User or an MNO.

Additionally, each component collecting and reporting information to the Ledger, through the Secure Oracle, in the form of a Verifiable Presentation (VP). These components that report information to the DLT include i) the Privacy-aware orchestrator reporting the Privacy Index, ii) the Security Probe reporting the attestation result, iii) the Proof of Transit (PoT) controller reporting the result from the

PoT, iv) the CTI reporting the threats and v) the SLAs as reported by the Orchestrator. In both cases (i.e., external and internal users and components) the entity that acts as the credential verifier is the SCB.

Lastly, PRIVATEER offers a **Trust Exposure Layer**, which guarantees the preservation of privacy, while allowing external entities, such as Mobile Network Operators (MNOs), users, or Service Providers to access specific data from the Ledger. More specifically, the Trust Exposure Layer ensures that the information provided to external entities is strictly limited to the trust state of a service, including information such as the trust property (i.e., integrity) along with the level of trust. Details, such as the exact evidence collected which may leak information regarding the underlying infrastructure, or service graph chain, are removed, protecting privacy without compromising the ability of external entities to assess the trustworthiness of the microservices. This harmonisation mechanism ensures that strictly relevant and necessary information is shared with external parties, harmonizing the information for safeguarding privacy; hence, effectively mitigating relevant risks.

2.1.2 PRIVATEER Flows

The PRIVATEER flow can be separated into phases. Please note that Secure Enrolment of the devices/assets is considered out-of-scope for PRIVATEER. Additionally, the Security Probe comes with pre-installed eBPFs and attestation capabilities; thus, in Release A of the PRIVATEER platform, it cannot be altered during runtime. If this stack requires modification, this means that the whole container has to be modified and relaunched.

2.1.2.1 Secure Launching of a Confidential Container, Verification of (Confidential) Container Workload & Container Binding

To ensure the proper operation of the edge-based framework and its services, it is essential that the containers are launched securely by the Orchestrator. It shall be noted that PRIVATEER seeks to adhere to the Confidential Container (CoCo) paradigm. In this regard, enclave-cc is utilized for launching confidential containers.

The Orchestrator determines whether to deploy a confidential or legacy container based on the specific security, service, privacy, and trust requirements provided by the Service Provider **(step 1)** (i.e., through an intent). These requirements reflect the service's needs, achieving a balance between performance and security. Evidently, not all containers should adhere to the same criteria. Therefore, these requirements serve as a basis for the service characterization. The service characterization process involves the translation of these high-level requirements into concrete security descriptions, which consider not only the service requirements but the infrastructure capabilities and imminent threats as well **(step 2)**. Insights regarding the threats arrive

to the Orchestrator from the Cyber Threat Intelligence (CTI) component, through the Security Orchestration, Automation, and Response (SOAR) system. For instance, based on these requirements and threat assessments, the Orchestrator may opt to deploy services on SGX-enabled servers.

Both legacy and CC-enabled image files are accommodated for deployment opportunities at the edge. Leveraging the Registry, the Orchestrator accesses containerized service images, network-oriented microservices and Confidential Computing (CC)-enabled services (e.g., Gramine TEE Overlay) for deployment **(step 3)**. The integration of a TEE Gramine Overlay is essential for deploying CC-enabled services, facilitating the creation of a service manifest file atop an .sgx version, generated through enclave-cc. This .sgx version establishes an overlay layer of the Gramine-enabled image on a standard image file, leveraging Intel SGX hardware to initiate confidential containers. PRIVATEER's adoption of Gramine as its TEE underscores its commitment to security, with "Gramine-enabled image" emblematic of the secure conversion of legacy containers.

The translation of requirements into characterizations culminates in the creation of an Interpretable Manifest, facilitated by the Manifest Interpretation component. This Manifest serves as a comprehensive set of configuration rules/actions, delineating the parameters necessary for deploying the microservice. It encompasses performance, security, trust, and privacy considerations, along with network and infrastructure capabilities **(steps 2 and 4)**. Within the Interpretable Manifest, details such as container size, bandwidth, network dependencies, and specific security requirements for CC-enabled images are specified.

Concurrently, the Orchestrator's SLA Manager formulates and transmits the Privacy SLA to the Service Provider, encompassing crucial information pertaining to performance, security, trust, privacy parameters, as well as network and infrastructure capabilities. This agreement entails all details that may facilitate the Service Provider in determining whether to proceed with deploying the service **(step 5)**. Upon agreeing to the deployment of the service, the Interpretable Manifest is transmitted to the Kubernetes Key Management System in order for the latter to initiate the secure launching of the container, leveraging the infrastructure assets **(step 6)**. It shall be noted that the Registry is enhanced with the images of the containerized applications as well as the Key Restriction Usage Policies. These policies do not solely refer both to the containerized applications but may further extend to the infrastructure assets **(step 7)**.

Typically, in the case of a legacy container, the Kubernetes key Management System releases the Kubernetes key upon deployment for authenticated communication with the Master Node, without any prior verification regarding the correctness of the deployed container. PRIVATEER provides an extension as it pertains to the Kubernetes Key Management, by offering verification of the container prior to the release of the

Kubernetes key and certificate. This verification is conducted on the edge side. The node receives the application manifest (i.e., either for a legacy or a gramine-enabled confidential container) and a challenge **(step 8)**.

The node, after deploying the service, calculates the MR Enclave Reference Value Measurement and signs it with the MR Enclave Key. Likewise, the challenge is signed by the deployed µProbe, leveraging its attestation key. This signature guarantees the container's authenticity, which is crucial in complex and multi-tenant environments where several instances of the same application may be instantiated; hence identification of the specific container that sends the attestation-related information is needed. The signed MR Enclave Reference Value Measurement along with the signed challenge is sent to the Kubernetes Key Management System, located in the Orchestrator for validation **(step 9)**. This attestation key may be generated prior or during the launching of the container. More details on the attestation are available in Chapter 3.

The Orchestrator knows the expected MR Enclave Reference Value Measurement for the specific node as well as the public part of the attestation key, hence it can validate the signature. If the received information is successfully verified, then the Kubernetes Key Management System releases the (Kubernetes) Key (i.e., asymmetric keypair) bound to the attestation key of the µProbe **(step 10)**. The presented certificate is bound to the container's unique attestation key, preventing unauthorized usage.

It shall be noted that a similar process is followed during the enrollment of the infrastructure element (i.e., SGX-enabled server) to the Orchestrator. More specifically the Security Probe residing in the SGX enabled sever is bound to the rest of the infrastructure (i.e., Orchestrator) through its own attestation key. This process is elaborated in Chapter 3.

Moreover, it shall be clarified that updating the Security Probe in Release A involves re-enrolment. In the Release B, the PRIVATEER platform will be enhanced with the capability to update the software stack of the infrastructure element without the need to re-enrol (i.e., which involves recreating the key to re-enrolling the infrastructure element). This feature could be specifically interesting for the case of inter-domain service continuity.

2.1.2.2 Definition of Trust Policy and update

After successfully deploying a service, the Orchestrator notifies both the Security Context Broker (SCB) and the LoT Assessment component **(step 11)**. This notification further includes information from the Orchestrator regarding the agreed Privacy SLA and the service graph chain topology. The latter is needed in order be aware of the exact trust sources that need to be queried in order to acquire the evidence. The LoT

Assessment utilizes this notification to craft a tailored Trust Policy for the deployed service. This Trust Policy encompasses various parameters necessary for assessing the trustworthiness of the service, including the privacy index, Proof of Transit (PoT), attestation results, and Cyber Threat Intelligence (CTI) data. Apart from the parameters, the Trust Policy includes information regarding the periodicity (i.e., how often to query the data) and the trust relationships (i.e., based on the service graph chain). This Trust Policy is sent to the SCB **(step 12)**.

The SCB is tasked with building a smart contract per service and defining the list of attributes. Nevertheless, the SCB which cannot directly upload a contract to the Ledger. This task is handled by the Security Probe. The Security Probe is the entity responsible for constructing and executing the (new) smart contracts. Hence, the smart contract including the Trust Policy, along with the rest of the information received from the Orchestrator such as the Privacy SLA and the service graph chain topology information is transmitted to the Ledger via Security Probe. The Orchestrator can use this Trust Policy-related information to initiate the attestation process. Should any updates be required to the Trust Policy, the LoT Assessment must accordingly update the smart contract.

2.1.2.3 Extraction of runtime attestation evidence from the Security Probe and µProbe for the LoT Assessment

The Orchestrator leverages information from the Ledger to determine whether attestation data is required for the trust assessment of a particular service, along with the periodicity at which this data should be obtained. This critical information is included within the Trust Policy field of the smart contract, providing clear guidelines for when and how attestation should be conducted to ensure the configuration integrity of the deployed services, the containers or even the underlying infrastructure. By accessing this data from the Ledger, the Orchestrator can effectively manage the attestation process; thus, maintaining the overall trustworthiness of the system.

The Orchestrator triggers the attestation process by sending a challenge to the Security Probe(s), that participate in the service graph chain for the specific service and more specifically its Attestation Agent(s) **(step 13)**. This communication is further specifying which binary should be attested. Along with the challenge, the Orchestrator sends the Key Restriction Usage Policy. The latter is leveraged by the Security Probe to verify the responses from the µProbe(s). Each Security Probe and µProbe that participates in the specific service graph chain reports separately.

In the example of Configuration Integrity Verification (CIV), the Attestation Agent residing in the Security Probe is tasked with verifying the integrity of the container. To accomplish this, the eBPF Tracer initiates the capturing of traces. The tracer consists

of two parts: one executed in the untrusted domain and another in the trusted domain. Hence, the traces are captured by the untrusted domain and signed by the trusted domain. Once the traces are collected and signed, they are transmitted to the Attestation Agent to initiate the integrity verification process.

In parallel, the Key Restriction Usage Policy Engine applies Key Restriction Usage Policy, which was configured during the enrolment of the infrastructure element. These Key Restriction Usage Policies seal the usage of the attestation key to a new configuration state. For example, the by the Key Restriction Usage Policy defines the attestation by proof (instead of attestation by quote) scheme, chosen for conforming with the Zero Trust notion. Moreover, the Verifiable Policy Enforcer (VPE), as part of the Key Restriction Usage Policy Engine (KRPE), monitors the status of the versions running on both the Attestation Agent and the Key Restriction Usage Policy Engine. It ensures that these components remain active and up to date, detecting any instances where they may have become obsolete.

Moreover, the Security Probe will send the challenge to the µProbe(s) in order for the later to verify the integrity of a specific binary, as it pertains to the containerised application configuration **(step 14)**. Following a similar rational, as performed in the Security Probe, the eBPF Tracer initiates the capturing of traces. Once the traces are collected and signed, they are transmitted to the Attestation Agent to initiate the integrity verification process. The result, meaning the signed challenge with the Attestation Key of the µProbe is sent to the Security Probe for verification **(step 15)**.

The result of the attestation process can be provided either by the Microservice Agility to the Secure Oracle or directly by the trusted part of the Security Probe through the Secure APIs. Nevertheless, the provision of such TEE Device Interface Security Protocol (TDISP) extensions is still under research by PRIVATEER.

The result of the entire attestation process, covering both the Security Probe and the µProbe, along the collected evidence is transmitted from the Security Probe to the Secure Oracle **(step 16)**. It shall be clarified that the collected evidence is available to the ledger specifically for the cases where attestation has failed. The Secure Oracle verifies the signature of the received information from the Security Probe and includes the attestation report to the smart contract dedicated to the specific service. All evidence is available to the Ledger in a common Verifiable Presentation (VP) format **(step 17)**. In instances where attestation fails, the raw information is stored off-chain for efficiency, with pointers to this external storage maintained on the Ledger. The LoT can then access the acquired information data through the Ledger to perform a new assessment **(step 18)**, leveraging the SCB. Details on the exact operation of the LoT are available in D4.1 [3] .

2.1.2.4 Attribute-based Access to the Ledger

Access to specific information on the ledger is regulated through Attribute-based Access Control (ABAC), ensuring that only entities possessing the correct attributes can access particular information. Furthermore, the information stored in the off-chain storage is encrypted using Attribute-based Encryption (ABE), providing protection against unauthorised access. The list of this attributes is defined by the Security Context Broker (SCB), who disseminates to the Secure Oracle in order to be included in the smart contract. The SCB further acts as the verifier, meaning that whenever an entity (i.e., either internal or external to the network) requests access to certain information leveraging a set of attributes that it possesses, it is the SCB that will verify whether the attributes possessed by the entity are indeed the correct ones to access the specific information. These attributes may be encapsulated within a Verifiable Presentation (VP).

2.1.2.5 Harmonization of Evidence for External Entities and Issuance of DIDs

The information exchanged in the Ledger (i.e., including the evidence available in the off-chain storage) may contain sensitive information about the infrastructure that should not be accessible to external entities. These entities could include end-users, other Mobile Network Operators (MNOs), or even the Service Provider that provided the service under assessment. To address this concern and provide an enhanced privacy-preserving solution, PRIVATEER introduces the Trust Exposure Layer. This layer implements harmonization mechanisms to conceal information that is not relevant for external entities, thereby preventing the leakage of crucial infrastructure details.

The Trust Exposure Layer serves as a protective barrier, controlling the information exposed to external entities accessing data from the Ledger. Its role is to ensure that only essential and non-sensitive information is made accessible. For instance, details pertaining to the service graph chain should remain private and not be disclosed to external parties. Instead, a summarized trust score reflecting the entirety of the service graph chain should be provided, offering a comprehensive yet abstracted view of the trustworthiness of the services involved. Through the Trust Exposure Layer, PRIVATEER maintains a delicate balance between transparency and privacy, allowing external entities to access essential information for trust-related decision-making while safeguarding sensitive infrastructure details **(step 19)**. This privacy-preserving mechanism further enhances confidence in the system among users and stakeholders.

To access this information, external entities must have previously obtained a Decentralized Identifier (DID) issued through the Ledger, specifically dedicated to Identity Management. This DID serves as the foundation for creating a Verifiable Credential (VC), which is then used to generate a Verifiable Presentation (VP). The VP

encapsulates the necessary attributes required to access specific information related to trust within the Ledger. By leveraging these cryptographic mechanisms, entities can securely present their credentials and prove possession of the correct access rights, without disclosing any further information, thus maintaining the sovereignty of their identity. The VCs and VPs are stored within the entity's wallet.

2.2 User Roles

In the context of this deliverable, it is essential to define the set of user roles that are provisioned so as to flesh out the necessary functional specifications of WP5 activities and facilitate the description of each artefact in the upcoming sections. The main user roles presented in this deliverable are presented below:

- Mobile Network Operator (i.e., Infrastructure Provider): These are the organizations responsible for the orchestration and management of the virtualized infrastructure where various services will be deployed. They further decide on the type of security controls and built-in security capabilities of them infrastructure and employ orchestration techniques for the optimal deployment strategy of all services to not violate any requirements as described in the Service level Agreements (SLAs).
- Service Provider: These are organizations that are responsible for offering the b5G/6G services, to be deployed as part of the underlying software stack, but also the auxiliary processes (deployed on a virtualized infrastructure such as the MEC) for supporting the better and more scalable execution of a specific service. The role of a Service Provider can also be taken by the developer of a security analytics solution (e.g., such as the AI-based anomaly detection from WP3) for equipping the PRIVATEER framework with runtime risk indicator capabilities that facilitate the decision-making process of the orchestrator with respect to the deployment of a service graph chain.
- End users: As end-users engage with various services and applications, their user equipment (UE) interacts with the network, leveraging Network Exposure Functions (NEFs) to access crucial insights into the capabilities and summarized trustworthiness data of service providers and underlying resources. For instance, when an end-user seeks to connect to a specific service, their UE may query NEFs to obtain trust-related data, such as the resources or security measures implemented within the service chain.

2.3 Functional specifications

This specification introduces the functional specifications in the context of the WP5 activities. These specifications stem from the requirements that need to be satisfied from the various actors – namely, Mobile Network Operators, Service Providers, End

Users etc. In addition, this table presents the component-oriented functional specifications which, in turn, are further broken down into sub-specifications stemming from the associated internal components of each artefact. Specifically, the first 6 specifications (i.e., F.S.1-F.S.7) characterize the user-oriented functional specifications. Subsequently, F.S.8-F.S.13 refer to the specifications related to the Privacy-aware Orchestrator, while the next three (i.e., F.S.14, F.S.15, and F.S.16) capture the LoT assessment requirements. It needs to be noted that for these artefacts, only the WP5-related specifications have been mentioned. Next, a detailed analysis of all the specifications pertaining to the WP5 tasks is presented: from the PRIVATEER DLT (i.e., Security Context Broker specifications: F.S.17-F.S.18, Secure Oracle: F.S.19-F.S.21, and Trust Exposure Layer: F.S.37), up to the Identity Management component (i.e., F.S.22-F.S.24), the CTI sharing (i.e., F.S.25-F.S.27), and the attestation mechanisms: both in the far edge devices (i.e., F.S.28-F.S.30), but also in the data/network plane (i.e., F.S.31-F.S.36).

Table 1 - PRIVATEER Distributed attestation, identity and threat sharing enablers Functional Specifications

ID	As a(n) ...	I want to be able to ...	So that I can ...	Description
F.S.1	Service Provider / End user / Mobile Network Providers	securely store the necessary keys, certificates, and decentralized identifiers	Have access to secure IdM mechanisms.	Securely storing cryptographic keys, certificates, decentralized identifiers (DIDs), verifiable credentials (VCs), and other sensitive credentials is essential in PRIVATEER. The adoption of the Hyperledger ARIES wallet provides a comprehensive solution, ensuring confidentiality, integrity, and availability. By integrating with ARIES, the end users achieve robust protection, meeting privacy requirements and instilling trust among stakeholders.
F.S.2	Service Provider / End user / Mobile Network Providers	provide verifiable evidence on my identity in a privacy preserving manner	get secure access into aggregated trust summaries	The user/service provider should leverage Verifiable Credentials to disclose proof of ownership of the necessary attributes that grants access to the information stored to the DLT. In this flow, the Secure Context Broker (SCB) authenticates and authorizes the actors, who provide a W3C Verifiable Presentation (VP) with the Identification Management component-issued attributes that they want to disclose to the SCB. The SCB acts as a verifier and checks for the validity of the W3C VP - including its revocation status. Subsequently, a fine-grained attribute-based access control mechanism is used to validate whether the request is authorized (i.e., has the correct attributes) to access the requested information from the DLT.
F.S.3	Service Provider	provide the service container images, configuration and service requirements	enable the orchestrator to deploy the service in the MNO infrastructure according to the service level agreement (SLA)	A manifest encompasses broader details about the deployment, operation, and management of a service within a B5G/6G network. It associates the service requested by the Service Provider with the characteristics of the MNO infrastructure resources. Service container images are included in the orchestrator's registry by the related service providers. If instructed by the service requirements, the service container images are launched in secure and isolated enclave processes once their correct state is verified. These interpretable manifests allow orchestrators to deploy the service using the required resources. This includes the decision on whether the service (or part of it) should be deployed in a TEE-enabled environment where containerized applications can leverage the capabilities of the µProbe. This implies that the applications are deployed in infrastructure elements that possess a Security Probe to perform the necessary attestation mechanisms and secure software updates. Finally, through these interpretable manifests, the orchestrator can derive the required level of assurance

ID	As a(n) ...	I want to be able to ...	So that I can ...	Description
				to be attained by the service graph chain. This information is also used to select the necessary key restriction usage policies to be enforced by the assets comprising the service topology. The enforcement of the key restriction usage policies is intrinsically linked with the necessary attestation tasks that should be supported by the service topology to monitor the Level of Assurance. The enforcement and evaluation of the key restriction usage policies and - consequently – the attestation tasks are carried out by the orchestrator.
F.S.4	Service Provider	have access to an immutable version of the SLA associated with the associated, deployed service	ensure that its clauses and conditions cannot be tampered with and enable all B5G/6G components have access to a common ground truth.	Once a service is deployed by an orchestrator, the agreed SLA between the Service Provider and the respective MNO is published to the DLT. Specifically, for each deployed service the orchestrator triggers – through the SCB - the creation of dedicated smart contracts responsible for expressing the SLA in an immutable and auditable way. This enables other PRIVATEER components – and authorized B5G/6G components – to access the SLA so as to perform their operation. For instance, the access to the SLA is essential for carrying out the LoT estimation as it provides all the necessary information with respect to the required trustworthiness level that a service needs to attain. For this purpose, it is crucial to have an immutable version of the Service Level Agreement to avoid any tampering.
F.S.5	Service Provider	monitor the aggregated trust summary of the service graph chain (through exposed trust summaries)	measure and verify the SSLA-Trust conformance.	Service providers have access to the respective aggregated trust summary pertaining to their service graph chains. Through their interface with the SCB, authorized service providers can consume trust state information reported in the DLT, without disclosing anything about the MNO's topology. Apart from the service providers, trust information is also consumed by the respective orchestrators so as to make the necessary decisions for ensuring the adherence of the SLA. In the context of the LoT estimation, this can be translated into ensuring that the actual level of trust is greater than the minimum accepted level of trust (i.e., required level of trust). Eventually, this could result in the construction of a new interpretable manifest by the orchestrator to respond (e.g., update the deployment) to the new LoT estimation results. Such a reaction might be related to the enforcement of additional security controls to the service graph chain to ensure the trustworthiness of the corresponding service.

ID	As a(n) ...	I want to be able to ...	So that I can ...	Description
F.S.6	Mobile Network Operator (MNO)	access inter-domain trust-related information	evaluate the trustworthiness of another domain without breaching its privacy	In the context of a multi-domain service graph chain, it is essential that participating MNOs can acquire aggregated trust-related information about each other's infrastructure. In this context, it is critical that the exchange of information is realized in a privacy-preserving manner – i.e., without disclosing anything about the actual intra-domain topology. To address this issue the plan is to build on top of existing standardized approaches for exposing network capabilities, such us through the definition of Network Exposure Functions [4]. Following the paradigm of the use of Application-Layer Traffic Optimization (ALTO) [5] protocol as a Network Exposure Function [6], the plan is to evaluate the extension of a NEF towards exposing aggregated trust results associated with the underlying network. This responsibility is carried out by the Trust Exposure Layer which is part of the Privateer DLT. Hence, in scope of this functionality, harmonization techniques are to be examined to ensure that no topology information is disclosed along with the trust guarantees of a domain.
F.S.7	Mobile Network Operator (MNO)	provide trust guarantees about the MNO infrastructure	participate in service graph chains spanned across multiple domains (service continuity)	As an MNO I need to expose capabilities and events to third-party authenticated and authorized Application Functions (AF). This would enable the MNO to host services that might span across multiple domains as instructed by the Service Provider's requirements. PRIVATEER envisions to enhance the information shared with NEFs so as to include aggregated trust information in a privacy-preserving manner. The trust results are reported by the Level of Trust (LoT) estimation of a service while the harmonization is carried out by the Trust Exposure Layer. For enabling the LoT estimation, the MNO needs to provision the collection of trustworthiness evidence. Such evidence may be static ones such as the verification of the correct configuration of the nodes comprising a service graph chain, but also dynamic evidence collected during runtime. The latter type, involves among others: Proof of Transit evidence, attestation evidence, Cyber-Threat Intelligence sharing.
F.S.8	Privacy-aware Orchestrator	certify the infrastructure elements and spawned containers	provide proofs pertaining to the secure launch of the resources that host a specific service	Each service graph chain is deployed in a set of containers (i.e., application containers NFVs) deployed within infrastructure elements. To be able to ensure that the service has been securely deployed in this environment, a set of guarantees need to be made beforehand so as to ensure that both the infrastructure elements (e.g., server node added in the MNO's cluster of resources) and the spawned

ID	As a(n) ...	I want to be able to ...	So that I can ...	Description
		participating in a service graph chain		containerized applications have been securely launched. Concerning the infrastructure elements that need to deploy TEE-enabled workloads, they are equipped with a Security Probe, while the respective containerized applications are deployed with their own µProbe. These probe components provide the attestation mechanisms to securely attest to the correct state of the associated entities - infrastructure element or containerized application - both during initialization (e.g., secure bootup), but also during runtime phase (e.g., configuration integrity verification). For these attestation mechanisms to be accepted a set of requirements need to be attained during initialization of the elements. First and foremost, the attestation mechanisms require a set of attestation keys that are securely stored and used by each infrastructure element and containerized application. These attestation keys need to be certified by the orchestrator so as to ensure the authenticity of the associated attestation data and to provide a proof that the orchestrator has verified the correct launch of each Security Probe and µProbe. Section 3 describes in detail all the necessary details pertaining to the setup and execution of the PRIVATEER attestation schemes.
F.S.9	Privacy-aware Orchestrator	have re-programmability capabilities in the available resources	manage the supported security controls that can be employed in the underlying infrastructure	The feature of re-programmability is of paramount importance in the parameterization of the infrastructure, especially in the case of configuring the security controls deployed in the infrastructure. This configuration of the resources is achieved through Security Probe(s), and specifically through the Microservice Agility component of the Security Probe(s) (see Figure 1). This allows the orchestrator to configure the security controls supported by the deployed containerized applications. Of course, in the event of an update in the containerized applications, the container needs to be re-launched to put the new key restriction usage policies in effect. Two main examples of such configurations with respect to the enforced security control consist of: i) upgrading the security controls for bug fixing and/or introducing new attestation capabilities, and ii) installing new eBPF functionalities for capturing new threats through the collection of specific application/container traces. Regarding the former type, enhanced security mechanisms (e.g., monotonic counters) are going to be evaluated to ensure that no rollback of updated security controls can take place to downgrade the security

ID	As a(n) ...	I want to be able to ...	So that I can ...	Description
				posture of a microservice. Finally, through the Microservice Agility layer it is possible for the orchestrator to trigger the collection of attestation evidence to monitor the Level of Assurance of the service graph chain, as dictated by the service requirements.
F.S.10	Privacy-aware Orchestrator	launch workloads and containerized applications across multiple resources in an isolated and secure manner	ensure the confidential execution of the deployed service graph chains	The use of confidential containers addresses this functional specification as they provide confidentiality and integrity for data, especially for runtime data. Confidential containers use hardware-based Trusted Execution Environments (HW-TEE) for resource isolation, data protection, and remote attestation. In the context of PRIVATEER, we leverage the enclave-cc project[4] which offers a process-based confidential container solution through Intel SGX [7]. The adoption of confidential computing lifts the trust assumptions that Service Providers need to have for the MNOs and the underlying infrastructure. Specifically, it is ensured that the client images are deployed in a confidential and tamper-evident manner (i.e., encrypted and/or signed images cannot be intercepted or corrupted by MNOs). In parallel, the workload data is managed in a confidential manner within the isolated environment (i.e., enclave process) without enabling MNOs to acquire access to the deployed data.
F.S.11	Privacy-aware Orchestrator	get an updated view of the trust state of the running edge services	make an informed decision about the deployment of the service graph chain in the underlying resources	The Orchestrator is responsible for the management of the correct deployment of the services as characterized by the SSLAs between the MNO and the corresponding Service Providers. Hence, they need to get as much insight as they can in order to make informed decisions about the optimal (i.e., in terms of resource management) and secure (i.e., in terms of threats in the infrastructure) usage of the underlying resources. The runtime assessment of the trustworthiness of a deployed service constitutes a crucial input that the orchestrator needs to consider when assessing the status of the service with respect to the SSLA. Hence, PRIVATEER should facilitate the adoption of a robust Level of Trust Assessment framework as presented in D4.1. On top of that, PRIVATEER needs to ensure the auditability and

[4] https://github.com/confidential-containers/enclave-cc

ID	As a(n) ...	I want to be able to ...	So that I can ...	Description
				traceability of such trust related information which adds additional trust guarantees to the orchestrator.
F.S.12	Privacy-aware Orchestrator	get real-time, trustworthy anomaly detection events from the edge accelerators	make an informed decision about the deployment of the service graph chain in the underlying resources	The AI-based security analytics inference operation takes place at the edge accelerator devices deployed in the MNO infrastructure. This enables the detection of any event that differs from the normal behaviour of the traffic within the domain. The use of such edge accelerators (e.g., FPGA boards) opens a set of attacks that need to be addressed to ensure the trustworthiness of the reported events. As further elaborated in Section 4 set of common attack vectors is mitigated using configuration integrity attestation mechanisms.
F.S.13	Privacy-aware Orchestrator	deploy the key restriction usage policies deployed in each containerized service (deployed in the edge)	enable local attestation schemes where provers can provide attestation evidence if they are at a correct state	Based on the manifest and the SSLA managed by the Orchestrator the necessary requirements for the level of trust are derived. These requirements are further translated into key restriction usage policies that need to be enforced in the assets of a service graph chain. This allows the infrastructure resources to attest to the correct state of their underlying Security Probe stack, to ensure that only provers with the expected set of components (e.g., eBPF tracer, Attestation agent, Verifiable Policy Enforcer as part of the Key Restriction Usage Policy Engine (KRPE)). The enforcement of such policies enables the adoption of a robust attestation scheme that enables verifiers to validate the correctness of provers' state in a zero-knowledge, distributed and scalable manner (e.g., Swarm Attestation scheme). In the context of PRIVATEER, the attestation schemes are configured by the Orchestrator. Once the Security Prove collects attestation evidence, these are forwarded to the Secure Oracle for verification before being sent to the DLT. From the prover's side the attestation process is configured via the Microservice Agility Layer which in turn triggers the Attestation Agent, residing in the trusted world of the Security Probe of each infrastructure element. The Attestation Agent is also responsible to collect – if requested – any attestation evidence coming from the deployed µProve in the containerized applications.
F.S.14	Level of Trust assessment	retrieve evidence from various trust sources in a	evaluate the expected and actual level of trust for the associated service and trust property (i.e., integrity)	PRIVATEER ensures the availability of evidence coming from all the trust sources, for the calculation of the actual level of trust of a service are provided. Even though trust sources may share their evidence directly with the LoT estimation, they could be also persisted through the Privateer DLT. This could enable traceability and

ID	As a(n) ...	I want to be able to ...	So that I can ...	Description
		verifiable and secure manner		auditability characteristics or even contribute to historical analysis by authorized entities. For instance, having trustworthiness evidence persisted to the DLT enables the construction of a reputation metric for the trustworthiness of a service. In parallel, having (failed) attestation evidence reported to the DLT may enable authorized entities (e.g., MNOs) to identify new vulnerabilities for the deployed services and resources.
F.S.15	Level of Trust assessment	compute and report the level of trust estimation for a specific service over a time window	Inform all interested parties about the trustworthiness of the deployed service	The LoT estimation computes information reported in a secure and verifiable manner through the monitoring service graph chain. This is accomplished through the Secure Oracle that is responsible for reporting all incoming trustworthiness evidence (e.g., attestation evidence) to the DLT. For a specific time, window, the LoT estimation collects all associated information for a specific service and calculates the actual trust level. This information is posted back to the Secure Oracle to amend the information uploaded in the smart contract. These level of trust results are then available to all authorized entities. On the one hand the orchestrator consumes this information from the DLT in the context of its decision-making process pertaining to the deployment of the service. On the other hand, end users, service providers and MNOs consume aggregated results of the LoT estimation to assess the trustworthiness of the underlying infrastructure and/or deployed service.
F.S.16	Level of Trust assessment	Keep track of evolution of the service graph chain for a particular service	enhance the information associated with a specific service in the context of the trust assessment estimations and for auditability purposes	Once the orchestrator has deployed a new service, the associated LoT assessment component gets notified about the deployed service graph chain. This enables construction of the necessary trust policies for the corresponding service, including the trustworthiness evidence that needs to be collected during runtime. Based on the decision-making process of the responsible orchestrator, the service graph chain is subject to change in the event of new functionality being included in the service or the re-evaluation of the deployment setup for performance and/or security reasons. This introduces the responsibility to the LoT assessment to update the enforced trust policies as well as ensure the association of existing information (e.g., trust results, trustworthiness evidence) with the new service graph chain. The LoT assessment is responsible for managing the update of the trust policies, but it is the Secure Oracle that should provide the necessary provenance metadata (i.e.,

ID	As a(n) ...	I want to be able to ...	So that I can ...	Description
				including the necessary pointers) so that the smart contract for the new service graph chain points to the existing information pertaining to the older service graph chains of the same service.
F.S.17	Secure Context Broker (SCB)	enforce access control policies for accessing the information stored on DLT	support the conditional access to data for entities that have the necessary attributes	In an intra-domain environment, the SCB provides the interface for Service Providers and MNOs to access the information stored to the DLT. Of course, only authenticated, and authorized entities can access the resources to the DLT. Specifically, the SCB authenticates requests that possess valid W3C VPs issued by the Identity Management component. From the W3C VP, the SCB extracts the presented attributes and uses its ABAC mechanism to decide whether the request has access to the intended data. Finally, as mentioned in Section 5, the PRIVATEER DLT distinguishes between on-chain and off-chain storage depending on the type of information. The decision of whether a piece of information is stored on-chain may depend on various factors, one of which is the size of the payload. In the case of attestation evidence, it might be optimal to store it off-chain and maintain the necessary integrity and authenticity checks on-chain to ensure that the off-chain data cannot be tampered with. Of course, on-chain and off-chain storage needs to be protected to ensure that only entities with the necessary attributes can access their data. For this purpose, the adoption of Attribute-Based Encryption mechanisms is considered in the context of the Secure Oracle functionalities (see F.S.17). The list of attributes used for the ABE mechanisms need to be stored unencrypted in a public channel.
F.S.18	Secure Context Broker (SCB)	receive notifications about the deployment of a new service	trigger the instantiation of the necessary smart contracts in the DLT	When a new service is deployed by the orchestrator, a notification is sent to both the SCB and the LoT Assessment. The latter creates the trust policy (e.g., set of trust sources, required trust level) and forwards it to the SCB. This set of information – i.e., the deployment notification and the trust policy – enable the SCB to initiate the creation of the necessary smart contracts to monitor the SLAs and trustworthiness level of the deployed server. More information pertaining to the smart contracts is reported in Section 5.
F.S.19	Secure Oracle	store trustworthiness evidence, SSLAs	ensure the integrity of the associated information	To avoid putting excessive trust to centralized entities, PRIVATEER envisions to leverage Distributed Ledger Technologies to deploy smart contracts for the execution and/or storage of trust-related information. For each service there is a

ID	As a(n) ...	I want to be able to ...	So that I can ...	Description
		and trust results, in an immutable, interoperable, and auditable way		dedicated smart contract for storing information such as the privacy SLA between the MNO and Service Provider, as well as for storing the LoT results. Trustworthiness evidence is also expected to be stored in the DLT, too. Since it is crucial to check the integrity of all the ingress data to the DLT, a Secure Oracle component is needed. Having a Secure Oracle (see Chapter 5) verifying the incoming evidence from various trust sources, increases the trustworthiness of the data to be used by the LoT estimation. Finally, when it comes to ensuring interoperability and verifiability of the reported evidence to the PRIVATEER DLT from the various trust sources, the use of W3C Verifiable Presentations is adopted. The aim is to express the reported trustworthiness claims in an interoperable fashion. The abstract data model for the expression of trustworthiness claims could be expressed using the IETF's standardized YANG data model specification [8], the details of which are to be presented in D5.2 [9].
F.S.20	Secure Oracle	encrypt information stored in the DLT	Protect the stored data from unauthorized access	When data are sent to the secure oracle to be stored in the DLT, it is possible that part of it may be stored off-chain. Such is the case of the trustworthiness evidence where it is appropriate to be persisted in an off-chain persistent storage, mainly for performance purposes. Hence the Secure Oracle provides the Attribute-Based Encryption (ABE) mechanisms to protect persisted payload. In addition to that, to ensure the integrity of the encrypted data, a pointer is stored in the smart contract (i.e., on-chain).
F.S.21	Secure Oracle	get real-time notifications about any changes in the Level of Assurance (LoA) regarding the service graph chain	get an updated view of the trustworthiness level of the deployed service	The Orchestrator is aware of the services' characteristics and requirements as expressed by the SLA. Hence, it is responsible for triggering – through the Microservice Agility layer - the Security Probe of those infrastructure elements where the intended service graph chain is deployed. The aim is to monitor specific trust properties (e.g., integrity) and measure the Level of Assurance (LoA) of the corresponding service graph chain. This monitoring is realized thanks to the Attestation Agent component deployed in the infrastructure elements (Security Probe) but also in the Attestation Agents in the µProbe (containerized application). This attestation evidence is sent to the Secure Oracle that verifies them and reports them to the DLT, should a change in the Level of Assurance is detected. This report is taken into account in in the context of the LoT estimation.

ID	As a(n) ...	I want to be able to ...	So that I can ...	Description
F.S.22	Identity Management component	provide a decentralized identity management solution in a privacy preserving manner	allow the authentication of service providers and other stakeholders to the PRIVATEER cloud infrastructure	Based on this requirement, PRIVATEER envisions to adopt the Hyperledger Indy [10] for the Identity Management Component. Hyperledger Indy is a Distributed Ledger Technology (DLT) specifically designed for DID management. Within Indy Hyperledger, it is possible to manage digital identities in a secure, privacy-preserving, and interoperable manner.
F.S.23	Identity Management component	issue verifiable credentials that showcase the identity and the attributes of an entity, while supporting revocation capabilities	enable the authentication and authorization of entities (e.g., service providers) to the PRIVATEER ecosystem	For the Identity Management component to function as an issuer of W3C Verifiable Credentials (VCs), it needs to possess robust capabilities in credential issuance and management. Hence, the use of Hyperledger Indy is adopted as a Decentralized IDentity Management component. This entails the ability to generate and sign VCs in accordance with the W3C standard, ensuring their validity and integrity. The Identity Management component should have a mechanism for securely storing private keys used for signing credentials, implementing best practices in cryptographic key management to prevent unauthorized access. Furthermore, it should support the creation and customization of credential schemas to define the structure and content of VCs issued by the system. Additionally, the Identity Management component should incorporate efficient revocation mechanisms to invalidate issued credentials promptly, enhancing security and trust within the ecosystem. With these comprehensive capabilities, the Identity Management component can effectively act as an issuer of W3C VCs, providing trusted and verifiable digital credentials while ensuring robust revocation processes.
F.S.24	Identity Management component	provide a client wallet for the verifiable credential holders that allows for the secure storage and selective disclosure of the associated attributes	ensure a secure authentication and authorization framework where all the necessary credentials are secure stored and presented in a privacy-preserving fashion	The adoption of Hyperledger ARIES wallet offers a powerful suite of capabilities crucial for secure credential management. The wallet provides a secure storage element for cryptographic keys and enabling seamless management of DIDs. Additionally, it facilitates the secure storage of verifiable credentials (VCs), ensuring their confidentiality and integrity. With ARIES, users can leverage selective disclosure verifiable presentations (VPs), enabling controlled and privacy-preserving sharing of credentials as needed.

ID	As a(n) ...	I want to be able to ...	So that I can ...	Description
F.S.25	CTI sharing entity	run in a decentralized mode	ensure that the CTI is highly available and does not constitute a single point of failure	The CTI Sharing proxy API must not rely entirely on one server for data exchange, and as such, each entity should host their own shared index which synchronizes with the other entities whenever a change happens.
F.S.26	CTI sharing entity	protect the cyber threat information collected from the various domains	control the access and ensure that only authorized users have access only to the intended parts of the CTI information	The CTI Sharing proxy API allows for better information control through the creation of these shared groups, which have different policies. This also entails the generation of a new shared secret key between all entities participating in the group whenever it is updated (someone joins or leaves).
F.S.27	CTI sharing entity	protect the identity of the domain of the reported CTI-related information	ensure confidentiality among the parties of the same shared group	The CTI Sharing proxy API relies on a reverse index to exchange information confidentially and securely. This entails being able to read, update and write to this index.
F.S.28	Edge accelerator device	have access to a secure element for storage, measurement, and reporting	support remote attestation operations for verifying the correct state of the device during initial configuration	Many efforts have focused on the development of Root of Trust provisions in low-end devices. To address such resource constraints, various proposals have focused on the inclusion of Physical Unclonable Functions (PUFs) to facilitate the secure generation of cryptographic keys to be used in scope of attestation protocols; even though it is not offering secure storage capabilities, it is possible to reconstruct the attestation key in a secure and unclonable fashion [11] [12]. In the context of PRIVATEER, the aim is to provide an integrity attestation protocol for ensuring that edge accelerator devices (i.e., FPGA devices) are loaded with the expected kernel applications in a confidential manner (see Section 5).
F.S.29	Edge accelerator device	to protect the FPGA bitstream deployed in the edge device	ensure confidentiality of the FPGA configuration information, especially in multi-tenant environment	Use of encryption techniques for application's bitstream, that contains the configuration for the hardware accelerators. This prevents unauthorized users accessing the code and thus mitigating reverse engineering attempts.
F.S.30	Edge accelerator device	acquire a certification about the application (i.e., AI accelerator	Avoid any malware updates and insertion of malicious circuits.	Remote attestation protocols are employed to verify the integrity of the users and the deployed code, as well as the attestation service running on the hardware accelerator.

ID	As a(n) ...	I want to be able to ...	So that I can ...	Description
		kernel) running on the device as well as the attestation service		
F.S.31	Attestation Server in edge accelerator devices	Obtain the necessary reference (expected) values for the application and device configuration running on the edge accelerators	Verify the attestation evidence reported by the edge accelerator devices	Regarding the integrity of the application, the reference values are uploaded by the developer of the accelerator to the attestation server, prior to uploading the application to the hardware accelerator
F.S.32	Attestation Service in edge accelerator devices	securely extract fresh attestation evidence	attest to the correct state (device and application) of the edge accelerator device	Responsible for generating the attestation report. Random nonces as well as PUF responses are going to be utilized, to enhance the security by providing robust encryption keys and avoid replay attacks. It is also important to verify the integrity of the attestation service.
F.S.33	Attestation Agent in containerized application and infrastructure elements	capture static and dynamic attestation evidence	provide attestation evidence to prove the correct state of the application both during design phase (i.e., bootup), but also during runtime	The attestation agent is a crucial component for both the Security Probe and the µProbe. It provides the prover's entry point for participating to attestation protocols and proving its correct state. It is responsible for the collection of the necessary attestation evidence – e.g., through the invocation of the corresponding tracing mechanism – as well as the invocation of the policy enforcement checks to locally attest to the correct state and, eventually, sign the attestation evidence. In PRIVATEER a robust configuration integrity verification mechanism is presented that allows containerized applications to provide runtime guarantees about the integrity and the correct configuration of the underlying system. To provide robust and scalable attestation framework, the µProbe Attestation Agents report their attestation evidence to the respective Security Probe Attestation Agent of the infrastructure element where they are deployed. Subsequently, once the Security Probe verifies the collected µProbe traces, it includes its own traces and reports them to the verifier located at the Secure Oracle. Another challenge that is

ID	As a(n) ...	I want to be able to ...	So that I can ...	Description
				investigated in the context of PRIVATEER is the ability to bind the attestation evidence with a specific enclave-cc instance of a containerized application (see Section 2.1.2.1). This is crucial in the context of B5G/6G landscape, especially given the replicated deployment of the same containerized applications (i.e., Kubernetes pod) across multiple resources.
F.S.34	Attestation Agent in containerized application and infrastructure elements	provide zero-knowledge evidence about the configuration integrity of the deployed application and the container	attest to the correct configuration of the prover entity without disclosing any type of the attested attributes.	One crucial performance challenge of the configuration integrity attestation schemes is the size of the configuration information that needs to be shared with the verifier. On top of that, in the cases where the verifier is not necessarily trusted, this poses a privacy risk as the evidence collected in scope of attestation protocols may lead to software/firmware disclosure attacks. One key improvement that is employed in the context of the PRIVATEER attestation mechanisms (see Section 3), is to provide trustworthiness evidence in a zero-knowledge manner, while enabling the verifier to validate the correctness of the prover's state without getting information about the underlying deployment and configuration details. Of course, the reporting of the evidence to the DLT by the Security Probe Attestation Agent (i.e., through the Secure Oracle) should be performed in a verifiable and interoperable manner through the adoption of the W3C Verifiable Presentations.
F.S.35	Verifiable Key Restriction Usage Policy Enforcer in infrastructure elements	provide guarantees that the correct key restriction usage policy is enforced	enable the modification/update of policies while ensuring the obsoletion of older policies	Based on input coming from the Security Orchestration and Automation Response (SOAR) element and other sources, the orchestrator may trigger the update of the employed security controls in the underlying infrastructure. Such security controls may involve the update of the installed Security Probe software stack (e.g., binary of the Attestation Agent). This communication is achieved through the Microservice Agility component. Such an action leads to a change of the configuration of the TEE. As a result, the already-enforced Key Restriction Usage Policy may no longer be valid. Such policies provide enhanced authorization mechanisms to the Prover's Attestation Key if and only if the respective protection policies, that are deployed as part of the underlying Root of Trust (i.e., in this case Gramine and Intel SGX), are satisfied. As mentioned in Section 3, part of the research explorations is to evaluate the security control updates coming from inter-domain orchestrators in the context of service continuity. In such cases, the Security Probe, through the Verifiable Key

ID	As a(n) ...	I want to be able to ...	So that I can ...	Description
				Restriction Usage Policy Enforcer needs to ensure the validity of such requests and ensure that the correct key restriction usage policy is in place.
F.S.36	Tracer in containerized application and infrastructure elements	securely measure and report runtime evidence of the application to be assessed	attest to the correct state of the prover containerized application during runtime	The Attestation Tracer is an integral part of the PRIVATEER attestation mechanisms both in the Security Probe and the µProbe (i.e., lighter version of the Security Probe attestation tracer). It is invoked by the respective Attestation Agent element for collecting runtime traces which attest to the correct state of the container. In the context of PRIVATEER, we plan to leverage the eBPF tracing capabilities. Leveraging the re-programmability offered by the eBPF framework, it is possible to define the necessary tracing capabilities for capturing specific traces as demanded by the Orchestrator. Depending on the request for attestation evidence, an eBPF tracer could be implemented to provide Configuration Integrity Verification (CIV), Control Flow Integrity (CFI), or even Network Flow Attestation (NFA) related evidence. The development of eBPF tracer applications imply that the collection of traces is deployed in the untrusted host – i.e., not within the TEE. Nevertheless, the signing of the traces and the enforcement of the associated key restriction usage policies are executed within the trusted world to ensure the trustworthiness with respect to the key restriction usage policy enforcement and the signing of the attestation evidence on the prover's side. To minimize the threat vectors due to the collection of the traces in the untrusted world, a research exploration is envisioned so as to evaluate the development of a secure set of APIs that report attestation evidence from within the trusted world. This could be achieved through the extension of the TEE Device Interfaces as specified in the TEE Device Interface Security Protocol (TDISP) [13].
F.S.37	Trust Exposure Layer	harmonize the trust-related information concerning the underlying domain infrastructure	share to authorized MNOs and end users about trust-related information to inter-domain service graph chains in a privacy-preserving fashion	Providing network exposure functions (NEFs) for exposing network capabilities is a crucial – and standardized – aspect in the context of B5G/6G architectures. PRIVATEER envisions to enhance NEFs enabling the inclusion of trust guarantees as part of the exchanged information. However, this exchange is not possible to be performed directly from the DLT, since this opens a wide spectrum of privacy concerns – especially in multi-domain use cases. For this purpose, a specific layer, namely the Trust Exposure Layer, is envisioned to sit between the DLT and inter-domain authorized users (e.g., MNOs) in an effort to provide trust-related

ID	As a(n) ...	I want to be able to ...	So that I can ...	Description
				information in a privacy preserving. The Trust Exposure Layer, which is part of the Privateer DLT solution, should provide the necessary harmonization and aggregation mechanisms to ensure that the shared trust-related information cannot disclose any information about the exact topology of the underlying infrastructure.

3 Runtime Attestation for varying Levels of Assurance(s) in virtualised environments

To conform to the Zero Trust notion, as priorly described, the trustworthiness of virtualised functions, systems, and their associated services should be attained. According to the International Standardization Organization, trustworthiness is the "ability to meet stakeholders' expectations in a verifiable way". While integrity verification is a pivotal aspect of trustworthiness, it is important to note that it is just **one of the characteristics of trustworthiness**, as defined in ISO/IEC TS 5723:2022 [14]. Other characteristics may include accountability, accuracy, authenticity, availability, controllability, security, privacy, quality, reliability, resilience, robustness, safety, transparency, and usability, according to the stakeholder's/landscape's requirements. PRIVATEER considers integrity verification through attestation as one of the trustworthiness characteristics used for the trust assessment. More information on the characteristics and the quantification methodology followed by the PRIVATEER's trust assessment framework is available in D4.1 [3].

More specifically, the trustworthiness of a device can be established when it consistently operates in the expected manner for its intended purpose [15]. To ensure effective runtime configuration integrity verification, it is imperative to collect **accurate system measurements**, **during the actual operation**. These measurements serve as evidence, indicating whether any tampering has occurred compared to a predefined "correct state". Through this ongoing collection and comparison of the collected runtime measurements with the established baseline, potential threats can be promptly identified and addressed, allowing for proactive responses to security incidents. This evidence can be also defined as traces, while they are signed by the attestation key (AK).

3.1 State Of The Art

3.1.1 System Configuration Integrity Verification as a crucial enabler for trust assessment

Evidently, it is crucial for these traces/measurements to be collected and verified in a secure manner, providing to any requesting party verifiable and trustworthy evidence over the system's operation. To ensure this, such capabilities should be integrated into the underlying Root of Trust (RoT). The RoT, as defined by the Global Platform, serves

as a computing engine, code, and potentially data, all co-located on the same platform, providing essential security services. It has several types of implementations. More specifically, it can be supported by a Trusted Execution Environment (TEE) or an embedded Secure Element (eSE). Three types of RoT may exist in a trusted platform: i) a RoT for measurement (RTM), ii) a RoT for reporting (RTR) and iii) a RoT for storage (RTS). The baseline for a RoT is to support secure storage (i.e., RTS). To enable the runtime attestation of platform integrity, measurement capabilities (RTM) are indispensable for generating and collecting measurements, while storing and reporting capabilities (RTS and RTR) are further necessary to provide evidence regarding any potential tampering with these measurements [16].

Towards collecting and verifying evidence in a provenly secure manner, a Trusted Computing Base (TCB) may be leveraged, providing the RoT capabilities needed for **protecting data in transit and data in rest**. The TCB is essentially a set of computer system components tasked with ensuring security. It comprises various security mechanisms and processes, including hardware, software, and firmware-based ones, that enforce security policies and manage the system's lifecycle. These components, including secure boot processes, cryptographic modules, access control mechanisms, authentication systems, secure storage mechanisms, and integrity measurement mechanisms, work together to protect the system from unauthorized access, modification, or exploitation. It is imperative for the TCB to consistently adhere to expected behaviour to avoid jeopardizing the overall security of the target system. The Trusted Computing Group (TCG) has played a pivotal role in standardizing remote attestation protocols, ensuring interoperability and compatibility across different platforms and vendors.

The objective of PRIVATEER is to **establish a flexible TCB that may be expanded dynamically**. This is accomplished by utilizing a modern Trusted Execution Environment (TEE) that allows for the dynamic addition of components during runtime. These components are safeguarded by the TEE, hence enabling the expansion of the TCB. The TEE offers, in essence, a secure environment for critical services such as computations and safety-critical binaries, separating the “trusted” with the “untrusted” worlds of the host; thus, serving as a RoT. In addition to the expansion capabilities offered by the TEE, another rationale behind choosing it for executing certain tasks is the need for a **minimised TCB**. The TCB often includes the operating system and most of the hardware (e.g. memory and storage). Minimising the TCB can be achieved by reducing the TCB’s trust assumptions on software; thus, removing the operating system out of the TCB. A minimized TCB reduces the risk of software bugs and errors, which could potentially disrupt the operation of critical services; hence, the TEE ensures that critical binaries will be timely and securely executed within its trusted boundaries.

Furthermore, PRIVATEER's Attestation Framework expands its scope by verifying the correct configuration of the virtualised environments including VFs, VNFs where services are instantiated, in real-time, through local attestation guided by key restriction policies to support **privacy-preservation**. The proposed attestation scheme is based on the **zero-knowledge proof paradigm**, ensuring that the Prover can provide evidence of its correct configuration state **without revealing any identifiable information** to the Verifier. To prevent unintentional disclosure, the Prover sends a proof of correctness, including the fulfilment of **key restriction usage policies**, instead of the actual traces. This approach aims to provide privacy-preserving features for various application domains. The protocol is challenge-based, with the Verifier initiating the process by sending a nonce, and the Prover providing a signature using its confidential Attestation Key. The secure enrolment phase ensures privacy-related restrictions are considered from the initiation of operation, facilitating the provision of suitable key material and key restriction usage policies. These policies do not permit the usage of the key if the state is not correct. This approach aims to capture both security and privacy requirements for various application domains.

3.1.2 Attestation of Virtualised Infrastructure Configuration

Virtualization techniques and containerisation demonstrate growing popularity, particularly in cloud computing and 5G infrastructures, improving services by enhancing resource management, thereby optimizing speed, availability, and latency. Even though Trusted Computing has provided essential assurances for the secure execution of critical tasks, ensuring the integrity and trustworthiness of systems and data, the offered protection is not directly applicable to virtualised infrastructures and containers. It becomes evident that the security assurances provided by Trusted Computing should be extended to further protect virtualised infrastructures, which have become increasingly prevalent in modern computing environments. Such an extension would involve developing specialized runtime security controls for trust and security, tailored to the unique characteristics and requirements of virtualized infrastructures.

Recognizing this gap, efforts are underway to extend Trusted Computing's security assurances to virtualized environments. One notable initiative is the Cloud Native Computing Foundation's (CNCF) Confidential Containers (CoCo) project. CoCo leverages hardware platforms like Trusted Execution Environments (HW-TEE) to enable confidential computing in cloud-native environments, ensuring data security at the pod-level. The CoCo project leverages established and developing hardware security technologies including Intel SGX (Software Guard Extensions), Intel TDX, AMD SEV, and IBM Z Secure Execution, along with novel software frameworks, to safeguard data in use. Following this approach, unauthorized access or modification of applications and data is successfully prevented, providing resource isolation, data protection, and remote attestation. Confidential Containers enable protection against

tampering with the infrastructure where the container is instantiated, and executed, allowing both secure bootup and secure (runtime) service execution, based on the underlying RoT. These assurances further represent the infrastructure's capabilities based on security controls, enabling as a result the Level of Trust assessment for the specific virtualised infrastructure, as discussed in ISO/IEC TS 5723:2022 [14].

ETSI has provided a classification that could be leveraged for mapping the extracted traces from virtualised infrastructures to specific Levels of Assurance (LoA) [17]. The scale uses 0-5, with a higher number indicating higher trust. PRIVATEER will leverage this classification to determine the appropriate scaling for its virtualized infrastructure.

- **LoA 0**: refers to the state of lacking any sort of integrity verification.
- **LoA 1**: involves verifying the integrity of the local hardware and virtualization platform (hypervisor) using signatures throughout the boot process and application loading. No evidence of integrity is provided. The integrity status is determined based on the platform state once the boot and application load operations have completed.
- **LoA 2**: Enhancing LoA 1 by including the verification of the integrity of the hardware and virtualization platform through remote attestation. Boot time and application load time measurements are being examined.
- **LoA 3**: expands on LoA 2 by incorporating the verification of VNF software packages at the local level, using signatures. It is necessary to verify signatures for all packages that are loaded when the VNF starts up, as well as when new packages are loaded (i.e., during the VNFCI boot and VNFCI application loading).
- **LoA 4**: involves the inclusion of remote attestation for the of VNF software packages. VNFCI boot time measurements and application during runtime measurements should be used.
- **LoA 5**: In addition to LoA 4, this includes the remote integrity verification of the infrastructure network, as well as the virtualization layer and VNF software packages, during runtime.
    - **LoA5a** refers to the use of remote attestation for infrastructure network remote verification, in addition to the checks required by LoA 4.
    - **LoA5b** refers to the use of remote attestation for run-time integrity state remote verification of virtualisation layer and VNF software packages, in addition to the checks required by LoA4.

Although Confidential Containers offer a reliable foundation for securing containerized applications, it is important to acknowledge that assessing the security levels of containers presents challenges in virtualized environments.

Remote Attestation of containers has been suggested through various methodologies, including "Container- Linux Integrity Measurement Architecture (IMA)". This method evaluates the integrity of each container on a platform using container PCRs. However, it has limitations, such as not authenticating closed containers and facing challenges when a container is stopped or restarted. Privacy for container measurements relies on a shared secret between the host system and the Verifier, but the protocol for securely disseminating this confidential information is not explicitly specified. In [18] the Keylime framework is leveraged to support a modified IMA, demonstrating low latency and independence from containerization technology.

In [19] a solution is proposed to address the security monitoring needs of lightweight cloud infrastructures. This solution leverages remote attestation to verify the software integrity of cloud applications throughout their lifecycle. The solution employs widely used technologies and frameworks, such as the Linux Integrity Measurement Architecture (IMA), the OpenAttestation platform, and the Docker container engine.

It becomes evident that conventional security techniques such as remote attestation, which were originally developed for physical systems, may have constraints when used in completely virtualized environments. In addition, specific security solutions may not be easily accessible in lightweight virtualization configurations. Despite these challenges, lightweight virtualization is gaining attention due to its inherent flexibility and minimal overhead. Therefore, this topic continues to be an area of interest for ongoing research.

3.1.3 PRIVATEER's Innovation in Runtime Attestation

PRIVATEER introduces an innovative approach by adopting a local attestation scheme instead of traditional remote methods, alongside privacy-preserving techniques such as zero-knowledge proofs, specifically attestation by proof rather than attestation by quote. However, to effectively verify configuration integrity, the extracted traces must undergo validation against predefined reference values and policies. One challenge in this context is supporting policy updates while ensuring that the version running in the relevant components is the most recent. To address this challenge, PRIVATEER introduces a novel functionality called the Verifiable Policy Enforcer (VPE) as part of the Key Restriction Usage Policy Engine (KRPE), detailed further in Section 3.2.1. The scope of the runtime attestation scheme is to ensure the configuration integrity of the VFs and VNFs. PRIVATEER will further research container attestation methodologies.

On the implementation aspects, the core of this framework relies on the use of a Root of Trust (RoT) and more specifically a hardware-enabled Trusted Execution Environment (TEE) named Gramine[5]. Gramine uses the Intel SGX technology to

[5] https://gramineproject.io/

protect software running on untrusted hosts. The TEE allows for a minimised and extensible TCB, as previously explained, while Gramine offers a lightweight environment for supporting secure and isolated enclaves with ease of porting to different OSes, and process migration. Leveraging this TCB, PRIVATEER securely and verifiably collects infrastructure traces for attestation, with all critical security services anchored to the RoT instantiated within the Gramine TEE.

3.2 Protocol description

3.2.1 PRIVATEER Security Probe

The PRIVATEER's Security Probe in order to be instantiated, leverages the extension routines of the underlying TEE, which are designed to support: i) the Key Management System, ii) the Key Restriction Usage Policy Engine, iii) the Attestation Tracer, iv) the Attestation Agent.

i) **Key Management System**: responsible for the secure generation, storage, and management of cryptographic keys. Emulating similar key management systems used in well-established Root of Trust (RoT) i.e. TPMs, where the cryptographic keys are compliant with the NIST standards and enable enhanced authorization access mechanisms through Key Restriction Usage Policies.

ii) **Key Restriction Usage Policy Engine (KRPE)**: offers one of PRIVATEER's core novelties, enabling the local attestation. Local attestation offers the option to a Prover to attest to the integrity of its configuration and behavioural state to a Verifier without divulging implementation details. This is realized through the use of policy-restricted attestation keys, which exclusively generate signed attestation attributes when a node's compliance is confirmed by the local Attestation Agent. The KRPE supports a collection of logical equations formulated from these assertions. In this context, the KRPE processes hash digests of inputs via logical ports to determine the validity of a live Key Restriction Usage Policy. To seamlessly integrate the KRPE into PRIVATEER's Security Probe(s) and µProbe(s), it is conceptualized as a child process initiated by the device's Key Manager. This setup ensures that the KRPE operates entirely within the Trusted World (enclave), maintaining communication integrity between the Key Manager, which handles key authorization requests, and the KRPE itself. Moreover, PRIVATEER's Security Probes' and µProbes' configuration mandates that entity creators or administrators define the set of actions permissible before an action is classified as "completed."
The PRIVATEER's KRPE further supports the Verifiable Policy Enforcer (VPE) functionality. The VPE is responsible for validating the Key Restriction Policy Engine (KRPE)'s current version and preventing potential attackers from having

multiple obsolete policies active. It ensures that outdated policies are identified and characterized as obsolete, preventing unauthorized or outdated ones from affecting the system's security posture. It monitors the correctness of software versions executed by the Tracer and the Attestation Agent within the enclave. If these versions are successfully verified, the VPE authorizes the enforcement of a specific key restriction usage policy. Such authorization is feasible through the use of a signing key that is injected to the VPE component from the SCB. The injected VPE key is also bound with an appropriate Key Restriction Usage Policy, indicating that the VPE component is in a correct configuration state and that the Tracer and Attestation Agent are executing the expected software version.

iii) **Attestation Tracer:** It consists of two parts: one operating in the untrusted or normal world, and the other running in the trusted world where the Tracer's secret key is securely stored. In the untrusted world, the Tracer continuously monitors processes and routines executed within each container or device, fetching new traces to collect essential information for attestation methods utilized in PRIVATEER, ensuring integrity. Its primary function is to capture and calculate the hashes of configuration properties from safety-critical untrusted processes and routines, thereby aiding in integrity verification. Although the monitoring in the untrusted world falls outside the TCB, a portion of the Tracer's execution occurs in the trusted world, within the TCB. This protected part encompasses cryptographic operations, including decoding raw security measurements, calculating real-time configuration hashes, and generating digital signatures over the configuration hash using the secret key. Once collected, the traces are signed by the Tracer in the trusted world and transmitted to the Key Manager to facilitate necessary operations.

iv) **Attestation Agent (AA)**: The Attestation Agent exposes Trusted Execution Environment (TEE) Device Interfaces based on the TDISP protocol [13], which provide runtime system measurements capturing the current device's configuration and operational state. These interfaces ensure the integrity of monitored traces even in the case of a compromised host. The Attestation Agent's role in providing authentic traces and ensuring secure exchange of these measurements is fundamental to the overall security and trustworthiness of the system. As PRIVATEER moves towards a zero-trust architecture, the trust assumptions are minimized. Two types of interfaces exposed by the Attestation Agent are Hardened-TDIs and Softened-TDIs. Hardened TDIs enable secure interaction with the PRIVATEER's TCB, capturing the consumption of "TEE-assignable" resources and monitoring the integrity and authenticity of the Attestation Agent. Softened TDIs allow interaction with "non-TEE-assignable" processes, which do not need to possess the required security/trust capabilities but have a critical role in the overall system function. These untrusted processes mediate the communication of these events to the

(trusted) AA for securely executing the required functionality. Due to their updateable nature, they are not considered part of the Trusted Computing Base, and attestation mechanisms are employed to verify their integrity as part of the Attestation Agent software stack.

3.2.2 Service Lifecycle Management

As described in the previous chapters, the core objective of the PRIVATEER's attestation mechanisms is dual: i) to enable the Orchestrator to securely launch confidential containers, and ii) to enable the enrolled devices and containerised services to attest their correct configuration during runtime when requested through a deployed attestation policy. This attestation policy defines whether a Configuration Integrity Verification process (or another) will be initiated. In PRIVATEER only CIV processes are supported. This attestation policy can be defined by the Secure Oracle that receives and validates the attestation reports from the Security Probe and pushes them to the ledger.

It shall be noted that the secure launch of confidential containers further encapsulates two phases: i) the secure enrolment of the Security Probe to the infrastructure (i.e., Orchestrator) and ii) the secure launching of a service within the confidential container along with the µProbe which enables attestation and tracing. Additionally, it shall be clarified that PRIVATEER aims at attesting the (runtime) configuration integrity for both the containerised application/service as well as the containers and devices supporting the deployment of the services, offering a holistic validation framework.

The aforementioned objectives (i.e., secure launching and runtime attestation) are achieved through two different protocols namely: i) the Verifiable Policy Enforcement and ii) the Attestation by Proof. During the secure launch of a confidential container, the Orchestrator requests and verifies the creation of a restrained asymmetric Attestation Key (AK) pair within the Trusted World of the Security Probe offered by the underlying RoT (Intel SGX); thus, adhering to the Key Restriction Usage Policy as defined by the Orchestrator. The Attestation Key of each Security Probe gets certified by the Orchestrator and can be used for providing attestation evidence. Although these mechanisms ensure the secure deployment, the secure lifecycle manager is needed to ensure the runtime configuration integrity of the containerized service.

3.2.2.1 Elevating Secure Launch of Confidential Container through Secure Enrolment of a Security Probe and an µProbe to the Orchestrator and Verifiable Policy Enforcement

The first step, prior to the launch of a confidential container, is the secure enrollment of the Security Probe to the Infrastructure (i.e., Orchestrator).

More specifically, we propose a new functionality to be instantiated within the Security Probe, as an extension to the well-established enclave-cc. This extension removes one of the two attestation measurements (i.e., MRSigner) from the confidential container, as the corresponding manifest is self-signed by a random, untrusted key from the host (i.e., server) prior to launching. Aiming to render this scheme more secure, the PRIVATEER's extension scopes to adopt **Fuzzing techniques** for the Orchestrator to deploy a unique Attestation Agent per Security Probe, but without altering any of its functionalities. This way the measurement of the enclaved Attestation Agent is unique and only known by the Orchestrator.

*It should be noted that the binary fuzzing is **bound with the server's unique identification public key** and is reproduceable and reversible only by the Orchestrator. This way, the Orchestrator can attest the Attestation Agent of the Security Probe making sure that is launched correctly and proceeds with certifying the Security Probe's Attestation Public Key.*

For the proper functioning of the PRIVATEER MNO-based framework, the secure launch of the MNO containers is of immense importance. To achieve this, PRIVATEER leverages the Kubernetes and the enclave-cc technologies, thus ensuring the reliable deployment and management of the confidential containers. To meet the requirements that PRIVATEER has set and support verifiability on the Key Restriction Usage Policy that is enforced, we are introducing a more sophisticated mechanism.

Since the Security Probe is now launched (as thoroughly described in Section 2.1.2.1) and configured correctly, making it part of the PRIVATEER's Trusted Computing Base, we can now authorize it to sign the manifest files of each µProbe; thus, enabling both attestation measurements offered by gramine. To be more precise, the MRSigner now is the digest of a certified public key, which private counterpart resides within the PRIVATEER's Trusted Computing Base. This way we can ensure that each containerized µProbe hosts strictly up-to-date and authorized applications, either running under the umbrella of gramine or in the UnTrusted world. Due to this approach, the gramine-enabled applications are configured as expected and are no longer vulnerable to the enforcement of an obsolete key restriction usage policy or to a potential unauthorized software rollback.

3.2.2.2 PRIVATEER Runtime Configuration Integrity Verification (CIV)

PRIVATEER plans to employ several attestation mechanisms so as to provide verifiable evidence regarding the execution of runtime services that are hosted within the PRIVATEER infrastructure. Such an attestation mechanism is Configuration Integrity Verification (CIV), used to ensure the correctness of the configuration of any containerised service that is deployed from the Orchestrator. As each MNO infrastructure is instantiated by TEE enabled with high computational power servers,

we are presenting a Zero-Knowledge CIV scheme based on the notion **Attestation by Proof.** More specifically, we are introducing a challenge response protocol, where a Verifier challenges the Prover with a fresh Nonce; if the Prover is able to handle the Verifier's challenge (i.e. is capable of producing a valid signature with its certified Attestation Key), then the Verifier knows that the Prover is in a correct configuration state.

By creating a valid signature, the Prover provides Attestation Evidence in a Zero-Knowledge manner, as he is not disclosing the actual configuration of the attested container. This is achieved by integrating the creation of the Attestation Key with the Secure Enrolment module, where the µProbe requests from the Orchestrator to issue a Key Restriction Usage Policy under which the Attestation Key will be bound.

3.2.3 CIV High-Level Overview

Having described all the entities and components that participate in the PRIVATEER's CIV scheme, we will now provide a high-level conceptual overview of the scheme. More specifically, it can be broken down into two distinct phases, namely i) Secure Enrolment and ii) runtime Attestation. During the Secure Enrolment phase, the Orchestrator sets up the appropriate Key Restriction Usage Policy, whereas in the runtime Attestation phase the attested µProbe gets challenged by the Security Probe to provide attestation evidence. Both flows are illustrated in Figure 2.

3.2.3.1 Secure Enrolment

1. Consider a new service owned by the MNO that needs to be deployed by the Orchestrator. Firstly, the service has to be deployed containerised as part of a confidential container following the Secure Launch protocol described in Chapter 2.
2. Afterwards, the Orchestrator makes the necessary actions for the creation and authorization of the Attestation Key. The Orchestrator fetches the reference values of the corresponding service in order to calculate the accepted configuration of the confidential container. In addition to the reference values of the containerised service, the reference values of the Attestation Agent Enclave (i.e., MRSigner and MREnclave) and a validation from the Tracer are appended for the computation of the accepted configuration. The expected configuration is signed and sent back to the Attestation Agents of the now enrolled µProbe.
3. The µProbe's Attestation Agent receives the signature computed by the Orchestrator, called Authorisation Ticket, and creates its Attestation Key bound with the respective (issued) key restriction usage policy. *It has to be noted here that upon creation of the Attestation Key (of the µProbe), its public*

part is sent to the Security Probe in order to be certified by the Security Probe's Attestation Key.

3.2.3.2 Runtime Attestation

Next, we describe the runtime attestation interface initiated by a remote Verifier (Security Probe), performed each time we need to collect Attestation Evidence from a µProbe, igniting basically a challenge response mechanism. Through this protocol the Prover creates evidence for the Verifier ensure that it is indeed in a correct configuration but without disclosing any information regarding its state

1. A Verifier, that is going to be the Security Probe, issues a fresh challenge that is going to be sent to the Attestation Agent of the Prover (i.e., µProbe).
2. Upon reception of the challenge the Attestation Agent creates a fresh *nonce* that is sent to the Tracer.
3. The Tracer on his behalf starts introspecting the requested services and extracts security measurements regarding the configuration of the attested services. Upon collecting the Traces, it uses its secret Key (Tracer_priv) to calculate a digital signature over the extracted traces and the nonce issued by the Attestation Agent. Afterwards, the signature along with the Traces are going to be sent back to the Attestation Agent. Let H() represent the hash of an information. The signature (σ) is calculated as follows:

   $$\sigma = Sign(H(Traces||nonce), Tracer\_priv) \quad (1)$$

4. The Attestation Agent creates a fresh session and starts executing the policy enforcement algorithm in order to get access to its Attestation key. The is initiated with zeros, hence *RuntimePolicy=(00...00).*
   - *i.* Appends to the fresh session the measurement of the entity that signed his configuration (i.e. the Security Probe)
     $$RuntimePolicy=H(RuntimePolicy||MRSigner) \quad (2)$$
   - *ii.* Appends to the session the measurement of the enclave application that is instantiating the Attestation Agent
     $$RuntimePolicy=H(RuntimePolicy||MREnclave) \quad (3)$$
   - *iii.* Appends to the session the Security measurement that the Tracer extracted.
     $$RuntimePolicy=H(RuntimePolicy||Traces) \quad (4)$$
   - *iv.* Verifies the signature of the Tracer with the Tracer's certified and pre-shared public key (Tracer_pub).
     $$Verify(H(Traces||nonce), \sigma, Tracer\_pub) \quad (5)$$
     If the verification is completed successfully the Attestation computes the name of the Tracer's key, which is the hash digest of the public key, *Tracer_Name=H(Tracer_pub) and* appends it to the runtime policy,
     $$RuntimePolicy=H(RuntimePolicy||Tracer\_Name) \quad (6)$$

v. With the *RuntimePolicy,* the Attestation Agent will verify the Authorization Ticket that it acquired by the Orchestrator in the Secure Enrolment phase,

Verify(RuntimePolicy, AuthorizationTicket, Orchestrator_pub) (7)

If the verification is completed successfully, the *RuntimePolicy* is re-set to:

RuntimePolicy=H(CC||Orchestrator_Name) (8)

where *CC* is the command code of a specific policy command.

vi. The Attestation Agent will recompute its AK_priv using the same KDF that was used during the Secure Enrolment phase. The newly derived key gets hashed and compared with the AK_Hash. If these two digests match, then the Attestation Agent has successfully recreated its AK_priv.

5. The Attestation Agent uses its AK_priv to sign the initial challenge the Verifier has sent to it and then discards the AK_priv.

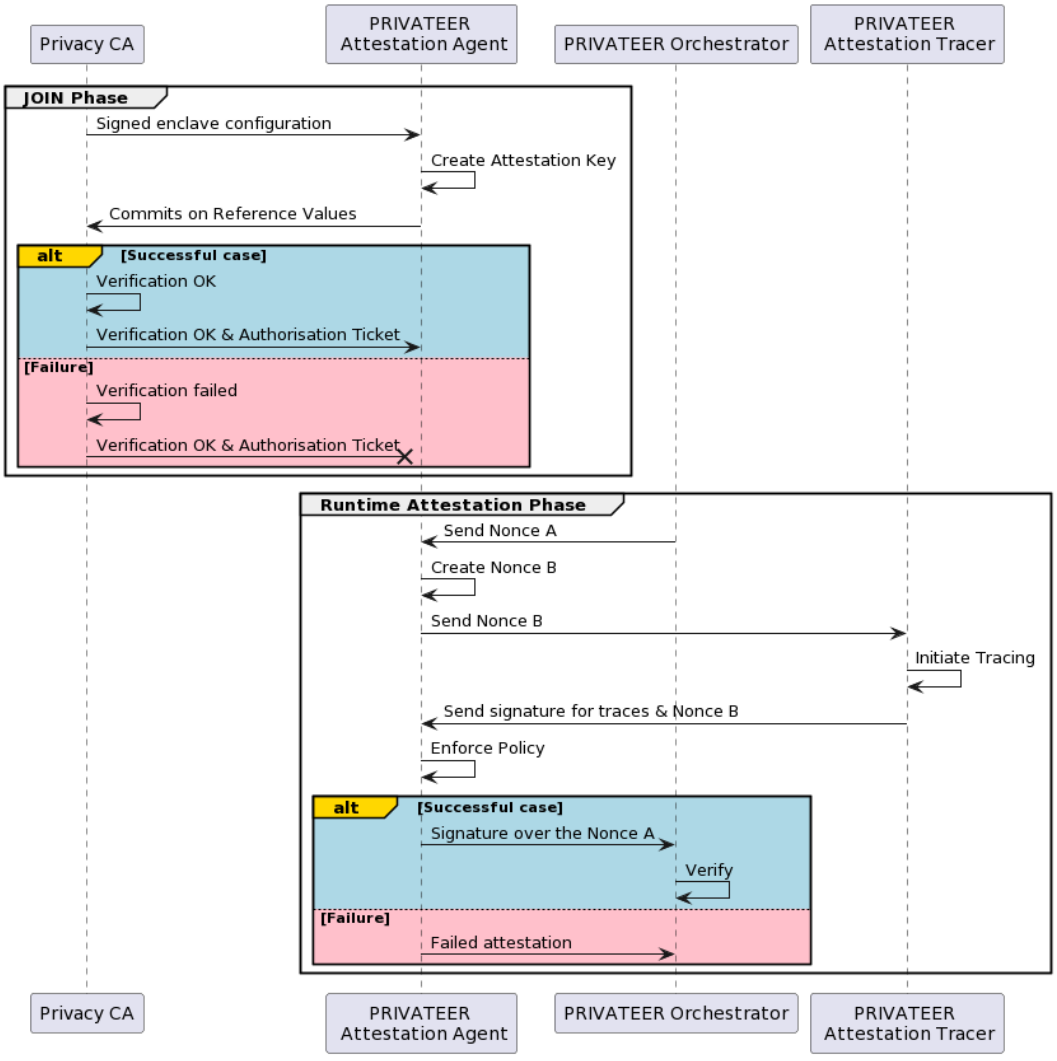

Figure 2 - PRIVATEER Join and Runtime Attestation Phase

3.3 Plan for development

In the present month within the domain of the PRIVATEER project, Task 5.1 has successfully instantiated a functional system designed to furnish a robust security framework tailored for cloud-based infrastructures. Moving forward, the trajectory of development entails a comprehensive exploration of supplementary methodologies and safeguards to fortify this system, capturing higher Levels of Assurance.

Primarily, regarding PRIVATEER's forthcoming Release A, efforts are concentrated on integrating Confidential Containers (CC) and advanced attestation mechanisms such as the described configuration integrity verification scheme presented in the above sections, into the foundational RA (Release A) protocol. To be more precise, in the current PRIVATEER framework release we have achieved to implement security mechanisms that achieve LoA 2. That being said, further development and design plans after the release A include the consideration of minimising the Trust assumptions regarding the Trustworthiness of the host, in order to elevate the Level of Assurance even further.

To this end, we are investigating attestation mechanisms to enable the attestation of every server that belongs to the PRIVATEER infrastructure, introspecting every security related aspect from VNFs to hosted containers. Such mechanisms are targeted to be investigated through the eBPF technology that enables tracing in kernel level of syscalls, netcalls and others.

In summary, the development in the remaining months of the PRIVATEER's will continue to explore new technologies and solution to enable introspection of various aspects of the system that are critical for the secure lifecycle of the PRIVATEER's infrastructure. The upcoming security modules will be integrated seamlessly with the existing operational base design, and the final iteration of Task 5.1 will be delivered in PRIVATEER's Release B.

4 Attestation in edge accelerators

The future network generations promise higher speeds to meet the connectivity requirements, but simultaneously introduce new challenges, especially in terms of security and privacy. Hence, real-time threat analysis and mitigation mechanisms are needed to protect sensitive data and critical infrastructure. However, **existing general-purpose CPUs cannot always meet the performance criteria** necessary for robust and real-time security in 6G networks. Towards this direction, alternative platforms such as hardware accelerators and more specifically Field Programmable Gate Arrays (FPGAs) are being investigated. FPGAs provide energy-efficient and high-performance computing capabilities, utilizing configurable logic blocks for digital logic. Unlike fully customized solutions (i.e. ASICs), they can be **reconfigured** according to the desired application requirements. In addition to the programmable logic, FPGAs often include a hard processing system (ARM-based or x86-based), which is responsible for configuring the device and transferring data coming from different systems. Although FPGAs offer an appealing solution, **the PRIVATEER tracing and attestation mechanisms described in Chapter 3 are mainly designed for virtualized infrastructures; thus, tailored trust extensions need to be considered for the far edge and edge sites to further support FPGA devices.**

In parallel, despite their performance efficiency, **hardware accelerators may still introduce security and privacy concerns** [20]. Figure 3 presents some of the most popular attacks, along with some of the available countermeasures. The threats can be divided into two sections, regarding the system's architecture: i) *Single tenant* referring to one application running to an Edge accelerator and ii) *multi-tenant* attacks representing scenarios where there is resource sharing from multiple users and kernels. Among the different options presented, attackers often perform unauthorized updates to application code, causing malicious effects on their functionality. They program devices with malware code, insert malicious circuits (often referred to as hardware trojans) for distributed Denial-of-Service (DDoS) attacks, or steal sensitive information. Side channel attacks rely on power and timing analysis, while replay attacks exploit security vulnerabilities. Attackers also reverse engineer user code to extract valuable information, such as the application's functionality. These attacks can be successful through various methods.

Recent literature explores methods to **ensure secure remote configuration of hardware accelerators using FPGAs' inherent reconfigurability**, aiming at protecting both the devices and the supported applications from the various attacks. Towards this direction, **remote attestation** protocols may be utilised, where essentially a host confirms to a remote server (through a predefined protocol tailored for each use case), the hardware configuration. However, managing the storage and generation of cryptographic keys poses a challenge.

Non-volatile memory (NVM) is commonly used for storing cryptographic keys, but it is vulnerable to attacks like side-channel. **To reduce these risks, researchers have developed Physical Unclonable Functions (PUFs) specifically designed for FPGA devices** [21]. PUFs create unique, impossible-to-replicate responses, which serve as authentication keys. This method simplifies the process of establishing a unique ID/cryptographic key for a specific FPGA device and strengthens security measures against unauthorized access. Regarding multi-tenant scenarios where there is resource sharing in a single accelerator card, isolation techniques between the different kernels are applied, while also power obfuscation strategies are utilized, to avoid remote power analysis attacks that can extract sensitive information (i.e. cryptographic keys).

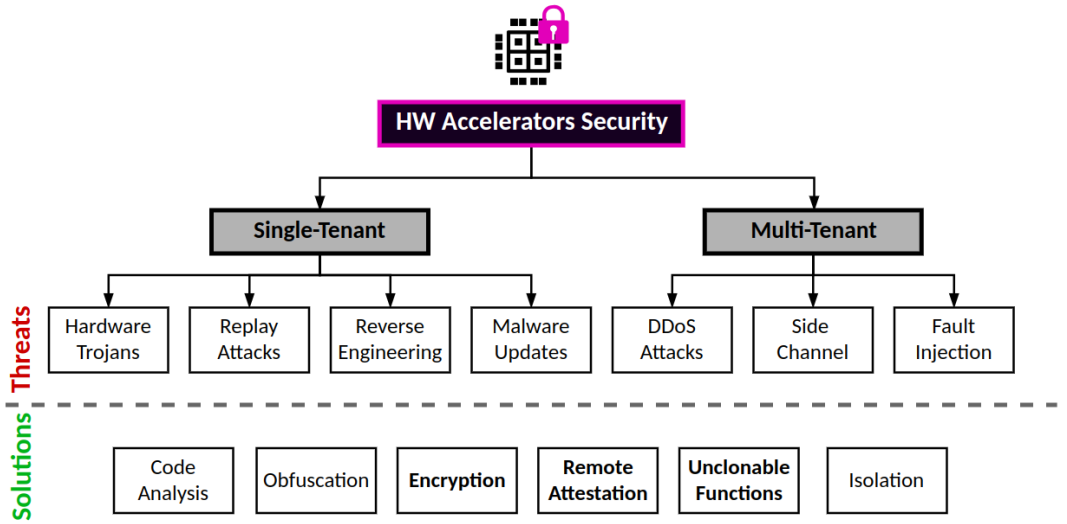

Figure 3 - Overview of hardware accelerators security attacks along with possible countermeasures

When examining different research works, it is important to also consider the trust models that are available for hardware accelerators. This should be accomplished based on the specific use cases of the application and the stakeholders involved [22]. The traditional trust model in hardware accelerator security comprises two main entities: i) the application developer and ii) their client, aiming to protect the designs for a specific target device. Towards protecting the design, encryption is performed, which binds it to a specific FPGA device that can decrypt it. Nevertheless, this method has specific constraints as it is confined to only one device and relies on implicit trust in the product creator. Consequently, it fails to meet the needs of contemporary applications and the different demands of stakeholders.

Hence, it is imperative to investigate alternative trust models that effectively manage security, flexibility, and trust, in order to promote innovation and resilience in FPGA-based systems within the ever-changing environment of 6G networks. An example of a trust model that should be considered in terms of PRIVATEER, is allowing developers to share their designs (i.e., AI/ML models for anomaly detection applications) with

multiple clients without revealing their source code. It is important to implement effective design rights management in order to protect the deployed core. An example of this nature typically entails the involvement of a trusted authority that facilitates the secure transfer of application cores to the FPGAs.

In the following paragraphs the State of the Art (SotA) is presented, proposing various methodologies to counter the aforementioned attacks with the respective security assumptions.

4.1 State Of The Art

4.1.1 Single tenant

Many research studies depend on the inclusion of supplementary modules alongside hardware accelerators at both the hardware and software levels to carry out secure configurations. At first, certain proposed schemes were based solely on **dedicated hardware modules of the target device**, without the involvement of a trusted third party (TTP). One example is described in paper [23], where the establishment of trust between the FPGA and the CPU is achieved through the **utilization of secure software that is linked to the bitstream of each application**. The clients are provided with an encrypted module and the corresponding software to manage CPU-based tasks.

“Fasten” [24] proposes an alternative method where users encrypt their designs specifically for a target FPGA device. The primary principle is based on the **utilization of pre-installed PUF-based public and private key generators** provided by the vendors. In order to mitigate the lack of trust with the platform provider, the user directly accesses distinct embedded keys to the FPGAs (which serve as unique device identities) from the vendor. The vendor maintains a secure database that records the deployed keys and their corresponding devices. Nevertheless, the absence of an external verification server renders these solutions inflexible and susceptible to vulnerabilities in the event of an update, while also raising concerns about user authentication.

More robust approaches are available in the literature that aim at enabling secure remote updates to devices through **remote attestation** protocols. One of the first schemes is proposed in [25], where one FPGA device securely receives an updated configuration bitstream from a dedicated update server. Each device is equipped with a unique identifier and cryptographic key, establishing a symmetrical key that is mutually shared between the FPGA and the server. The server employs this key to transmit encrypted bitstreams to specific FPGAs via an unsecured communication channel. In addition, the scheme includes a remote attestation mechanism that verifies the current configuration's status and the update process. The update server operates as a Trusted Third Party (TTP), with the responsibility of allocating

cryptographic keys and configuration bitstreams to individual Field-Programmable Gate Arrays (FPGAs).

A more recent work with additional functionalities is presented in [26] by the name "ShEF". ShEF is designed as a shield component that can be integrated into existing FPGAs to ensure that only authorized users can access the hardware accelerators in the cloud and maintain data confidentiality. Their solution **offers a secure boot procedure and a remote attestation process** to restrict access exclusively to authorized users. The establishment of RoT is achieved by utilizing dedicated embedded keys in the FPGA, while a software-based security kernel verifies the received acceleration module. In addition, hardware-based encryption modules are utilized to address the possibility of malicious users gaining access to data.

Furthermore, authors in [27] propose a scheme where self-attestation of the FPGA device is performed, namely **without a software-based process**. The FPGA is divided into two partitions, one static and one dynamic. The first remains unchanged for performing the verification, while the second one is reserved for loading the hardware accelerated application. The attestation protocol relies on securely erasing the existing memory contents in the FPGA prior to uploading a new application code; therefore, any potential pre-existing malicious modules are also removed from the device. Furthermore, the network traffic passes only through the FPGA device, where along with the key generation module, the required Ethernet core is implemented. Like the other works, the remote attestation process involves communicating with an external server, acting as a verifier, where the exchanged data is authenticated.

In addition to the aforementioned studies, various works focus on the **SoC-FPGAs which consist of an ARM-based CPU** in conjunction with FPGA logic. Most of the suggested methods for guaranteeing the integrity verification of Edge Devices rely on the existing ARM Trustzone TEE [6]. This trusted execution environment, designed for ARM CPUs, safeguards sensitive user code by offering **hardware-enhanced isolation** for these applications (as mentioned in Chapter 3). A recent study [28], implements modules for securely booting accelerator kernels in FPGAs alongside the existing TEE solutions. In addition to the in-device functionalities, an external proxy-server is used for performing a custom remote attestation protocol.

4.1.2 Multi-tenant

The use of **multi-tenant application of FPGAs**, where multiple kernels are implemented on a single device, is becoming increasingly popular due to the advancement of partial reconfiguration. This feature allows for the programming of specific sections of the device. Various research works in literature are focusing on countering side channel attacks. For example, authors in [29] propose a **power**

[6] https://www.arm.com/technologies/trustzone-for-cortex-a

obfuscation solution, in which using adjustable noise generating circuits in the FPGA, keep the power consumption of the accelerator constant, thus making significantly harder any power analysis attempts to extract sensitive data (i.e. encryption keys).

Different approaches are presented in [30], where a **routing-based** solution is utilized, that tries to reduce the long transmission lines between the different tenants in a single FPGA device, therefore minimizing the points for performing side channel attacks. To restrict the efforts of malicious users who attempt to insert circuits that carry out power-hammering operations, which can result in distributed denial-of-service (DDoS) attacks, researchers have created a protective software called FPGADefender [31]. This software scans the netlist of the application and identifies any modules that exhibit such behaviour.

4.1.3 PRIVATEER's Innovation in Edge Accelerator Attestation

Although various methodologies are available in the literature for securing hardware-based systems, in the current 5G infrastructure no such security countermeasures are applied. PRIVATEER aims to address this gap by offering robust methodologies to enhance the security of hardware accelerators in future networks. The proposed architecture provides a comprehensive approach that integrates various methodologies, allowing to securely configure FPGA accelerators. This will effectively protect user code and deployed systems from potential malicious attacks.

4.2 Protocol description

The core of the developed methodology for enhancing the security of the hardware accelerators, is based on a custom **Remote Attestation** protocol, with the abstract architecture of the system shown in Figure 4. The proposed protocol involves the interaction between three parties: i) the User, ii) the Attestation Server and iii) the Edge Accelerator. In relation to PRIVATEER, i) the User represents the *developer* of the hardware accelerators as described in Task 3.5, ii) the Attestation Server refers to an external verification server that may reside at the Orchestrator level, while iii) the Edge accelerator (i.e., FPGA) is being executed at the accelerator kernel, residing in the Infrastructure Layer (see Figure 1).
Note that AMD/Xilinx's FPGA devices can be divided into two categories, depending on their system's architecture: 1) MPSoC FPGAs, where an ARM processor is populated along with the FPGA logic; 2) ALVEO cards, where the FPGA logic is implemented on an expansion card that connects to a host computer via PCI-Express. Therefore, the device family of the Edge accelerator can be either a MPSoC FPGA, considering an edge computing platform is utilized, or an edge server where an ALVEO card will be employed.

In the hardware accelerator, a hybrid remote attestation mechanism is developed, where both the hard processing system (host x86 CPU or ARM-based processor), as well as the FPGA programmable logic is utilized for executing the various security-related functions. The network functionalities (communicating with the attestation server), as well as the bitstream loading is performed from the hard processor (i.e., PRIVATEERs Network Control Plane layer). On the other hand, the required encryption and key derivation modules can be implemented in the Programmable Logic (PL). Additionally, to verify the device's identity and achieve a robust RoT, an FPGA-based **PUF** can be utilized.

In this setup, the external verification server (which in terms of PRIVATEER resides within the boundaries of the Orchestrator) also plays the role of a TTP and is responsible for having a secure communication channel between the nodes in the system. Simultaneously, it has the role of managing the update of the reference values in case the user wants to perform an application upgrade.

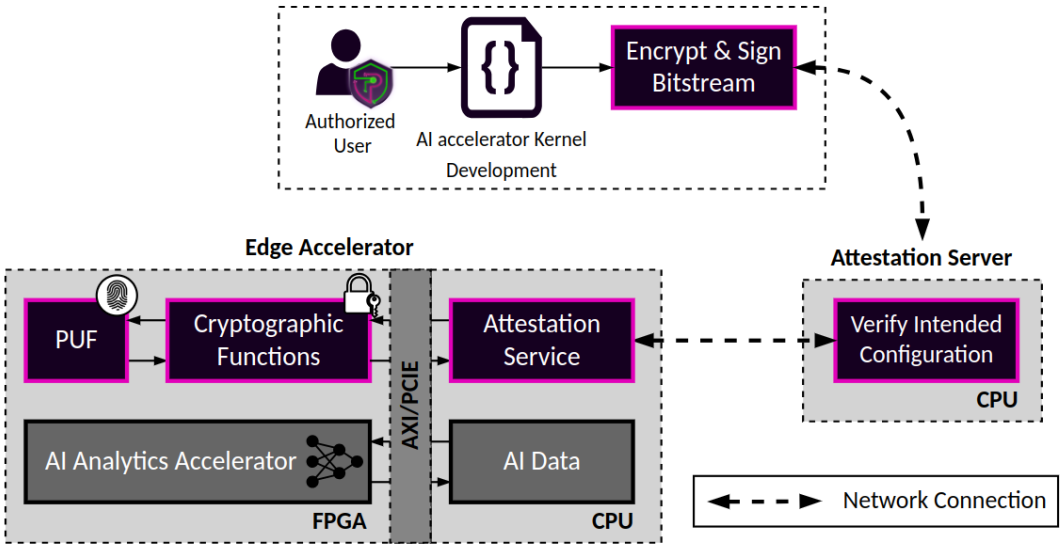

Figure 4 - Architecture of Edge Accelerator and external Attestation server setup

The detailed steps of the RA protocol are illustrated in Figure 5, while the main attestation procedure can be divided into three main stages:

- *Stage 1:* The *User* after developing the Accelerated kernel, produces the bitstream that contains the final's application configuration and is the file that will be loaded in the hardware accelerator. Then the user applies the required countermeasures to protect its design and communicates with the *Attestation Server* (Steps 1-2).
- *Stage 2:* The *Attestation Server* (residing within the Orchestrator) after establishing a secure connection and verifying the integrity of the attestation application running in the *Edge Accelerator*, proceeds with checking the authenticity of the uploaded application, as well as the FPGA device (Steps 3-8).

- *Stage 3:* In case of a successful verification of all the parameters (successful RA), the *Attestation Server* forwards to the Edge accelerator the bitstream decryption key required for loading the AI Accelerator program. (Steps 9-10).

Remote Attestation Protocol

Giving a more detailed analysis of the proposed remote attestation protocol illustrated in Figure 5, firstly an offline phase is executed, in order to collect the PUF responses from the available FPGA devices, which are then securely stored in the *Attestation Server* **(Step 1)**. This procedure is performed between the *Attestation Server* and the operator of the *Edge Accelerator*, which in PRIVATEER's use case can be identified as the developer. Additionally, included in the offline phase is obtaining the reference checksum value of the attestation service **(Step 1)**, which also must be performed only one time by the developer of Task 5.2, that is providing the hardware security infrastructure. Furthermore, we note that the collection of the PUF responses and the checksum of the security application is a one-time performed task (per FPGA device), and no recollection of these data is required in case the user wants to update a newer version of the accelerator.

As a next step, the user prior to uploading the application kernel to the Edge Accelerator, performs an encryption and signing of the bitstream with a unique key and the corresponding certificate, and finally transfers them to the Attestation Server **(Step 2)**. These correspond to the reference values that will later be used for verifying the integrity of the edge accelerator's code, which are denoted as "Golden Values".

After this preparation, the user is ready to make a request for RA. After establishing a secure connection with the Attestation Server and sending the corresponding request with a randomly generated nonce (N1) to avoid replay attacks **(Steps 3-4)**, the Attestation Server receives the checksum of the attestation application along with the transferred nonce, to verify that the correct verification service is running at the Edge Accelerator **(Step 5)**.

The Attestation Server then sends an attestation request to the Edge Accelerator along with a new randomly generated Nonce (N2) and the PUF challenge (PC) **(Steps 6-7)**. We note that prior to the request for RA, the user has already uploaded the encrypted application bitstream to the edge accelerator, to be ready for authentication. This transfer is performed using a generic secure file transfer protocol, using the already secure connection that has been established. Then, the edge accelerator calculates the static checksum of the received bitstream, as well as getting the included bitstream certificate to generate the AttestReport. This process can be performed from the FPGA host in software level (ARM CPU or x86 based CPU depending on the used device). Additionally, in order to avoid any potential sensitive information leakage, the checksum, the received certificate, and the received nonce are encrypted along with the PUF response in the FPGA. The generated attestation

report (AttReport) then gets transmitted to the External Server for verification **(Step 8)**. The encryption of the transmitted message is performed to also ensure the device's authenticity, since the PUF response is unique per device.

Then the Attestation server using the prestored PUF responses for the used FPGA device, decrypts the AttestReport to check if the transmitted variables match with the golden values initially received by the user. After a successful authentication of the AttestReport, the bitstream decryption key (BitstrDecKey) is forwarded from the User to the Edge Accelerator, using a secure key exchange algorithm (i.e. Elliptic Curve Diffie Hellman - ECDH) **(Steps 9-10)**. After receiving the DecrKey, the Edge Accelerator decrypts the user code and loads the application to the Hardware Accelerator.

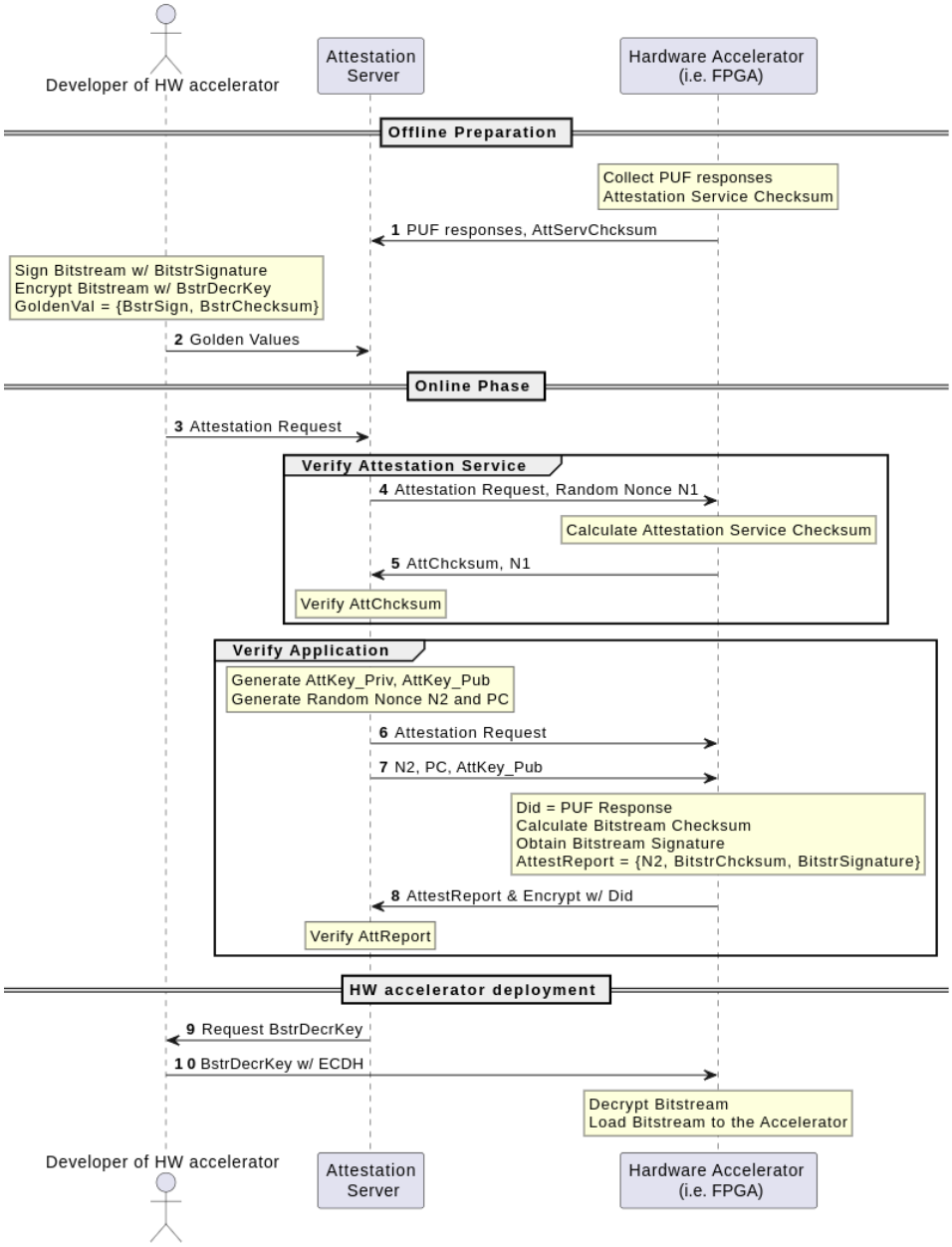

Figure 5 - Suggested RA protocol for the security of Hardware Accelerators

4.3 Plan for development

In the current month of PRIVATEER project, Task 5.2 has implemented a working system that provides a security infrastructure for hardware accelerators. During the remaining months, the development will continue, by exploring additional techniques and countermeasures which can be incorporated into the system. Firstly, regarding the PRIVATEER's Release A, the integration of PUFs and additional FPGA-based units (i.e. attestation report generation module) into the base RA protocol is being explored, along with evaluating different PUF implementation, to find the most suitable one. Additionally, further development plans after the Release A include the consideration of reducing the assumptions regarding the trustworthiness of the attestation server, in order to enhance the security of the system. This incorporates an additional node to the RA protocol, that could either be an external standard TTP service, or it can also be integrated in the PRIVATEER's Blockchain. Further security countermeasures that can be considered include enhancing the security of the software-based tasks, for example by using CPU-based TEEs for the software-based applications involved in the remote attestation, as well as researching countermeasures aimed at multi-tenant scenarios. Specifically, this refers to methodologies in the scenario where multiple AI accelerator kernels are running in the same FPGA card and thus resource sharing occurs.

To summarize, the development in the remaining months of PRIVATEER's will continue exploring additional techniques and solutions to mitigate hardware security attacks relevant to the project's threat model. The future developed security modules will be incorporated with the base design that is already available and operational, while the final version of Task's 5.2 will be delivered in PRIVATEER's Release B.

5 Blockchain for secure data exchange of trustworthiness evidence

The blockchain serves as a fundamental component for 6G networks, enabling intelligent resource management, spectrum sharing, scalability, and availability for emerging smart environments such as healthcare, smart cities, industry 4.0 and agriculture. The ability to **securely share, exchange and store data**, along with strong monitoring and traceability methods, makes it a crucial component of 6G technology, as explained in D2.1 [32]. By allowing tracking of data origins and exchanges within the network, blockchain enhances transparency; thus, trust.

Blockchain technology provides a compelling solution for auditability, efficient querying, and access control due to its unique features. One key advantage is its **immutable** record keeping, where data recorded on a blockchain cannot be altered or deleted once added. This characteristic ensures the **integrity and reliability** of the data, making it ideal for auditing and compliance purposes. Additionally, blockchain transactions are **transparent** and **decentralized**, enabling auditors to easily trace transactions and verify data authenticity across the distributed ledger.

Blockchain is a chain of records called blocks, linked and secured using cryptography. Each block contains transaction data, a time stamp, and the hash value of its previous block. These timestamps are leveraged for ensuring **the auditability of transactions,** linking each transaction to previous ones, allowing auditors to track the flow of assets or information over time accurately. In the blockchain each node, in this network of nodes, has a copy of the transaction records. This makes all records accessible and easily verifiable. Modifying a block requires consensus from all nodes, making it expensive for nodes to modify data. Blockchain technology allows untrusted parties to make transactions securely without the involvement of a central authority regulating them. Cryptocurrencies based on blockchain have gained attention, with smart contracts emerging as an evolving area.

In addition to the auditability, **encryption**, **integrity protection** mechanisms, **authentication**, and **access control mechanisms** are applied, restricting access to authorized entities, while ensuring accountability of actions. Smart contracts, which are self-executing contracts with predefined rules encoded on the blockchain, enable sophisticated access control mechanisms. These contracts can enforce access policies, specifying which parties have permission to access specific data or perform certain actions on the blockchain. This granular control enhances security and privacy, particularly in industries with stringent regulatory requirements like healthcare and

finance, where sensitive data must be carefully protected. Regarding the exact access control and encryption mechanisms, **Attribute Based Encryption (ABE)** and **Attribute Based Access Control (ABAC)** can be exploited within the blockchain setup to offer confidentiality and authentication linking while restricting access based on certain user, system, or device properties.

Moreover, blockchain technology is **device and type-agnostic**, providing flexibility by allowing access to devices with limited computational and storage capabilities. This enables the exploration of advanced functionalities of next-generation networks without the requirement to upgrade existing hardware. Simultaneously, it facilitates data portability, particularly when data owners desire to transfer their data from one blockchain ecosystem to another.

However, blockchain-enabled data sharing must not neglect **privacy** requirements. To comply with both security and privacy requirements, access control and confidentiality requirements should be considered. Towards this direction, distinctions between public and private ledgers must be performed according to use case scenarios, while enhanced crypto primitives could be employed according to the requirements of each application scenario. The variations among different ledger solutions are summarised as follows:

- **Public/Permissionless blockchains** are open-access networks that allow anyone to participate without prior approval, promoting decentralization. Examples include Bitcoin, which achieves consensus across an anonymous network. However, these blockchains often face scalability issues and slower transaction speeds due to the need for consensus mechanisms.
- **Private/Permissioned blockchains** restrict network participation to entities vetted through an authentication process, enhancing privacy and operational efficiency. Examples include Hyperledger Fabric, which supports various industrial applications and ensures the integrity of network participants.
- **Hybrid blockchains** combine elements from both public and private blockchains, offering a balanced approach that balances privacy and transparency. These blockchains allow selective participation, making them attractive for organizations seeking to capitalize on their potential.

Table 2 demonstrates a comparative analysis between the different types of Blockchain solutions (i.e., public, private and hybrid) in terms of access, authority, transaction speed, efficiency, data accessibility and immutability.

Table 2 - Blockchain types overview

Item	Public Blockchain	Private Blockchain	Hybrid Blockchain
Access	Open to all	By authentication only	Selective access

Authority	Fully decentralised	Centralised control elements	Combination of both
Transaction Speed **Consensus Mechanism**	Typically slower Open participation	Enhanced speed Restricted to members	Customisable Adaptable
Efficiency	Variable	Optimised for private networks	Tailored to requirements
Data Accessibility	Universal read/write	Limited to authorised Entities	Configurable
Immutability	Immutable	Conditionally im-mutable	Variable based on rules

5.1 State Of The Art

5.1.1 Smart Contracts

The term is popularly used to refer to low-level code scripts running on a blockchain platform [33]. Smart contracts are software programs that are recorded on a blockchain and are designed to execute automatically when certain preset criteria are fulfilled. Contracts like these became popular due to their ability to automate and optimize numerous operations, removing the requirement for intermediaries and guaranteeing prompt execution of agreements. Smart contracts are self-executing protocols that function autonomously within the blockchain network. They execute predetermined actions in response to specific situations and triggers, following a "if/when...then" structure. This architecture not only speeds up transactions but also fosters trust by allowing parties to immediately determine the outcome without the need for intermediaries.

The lifecycle of a smart contract involves several stages, from design and deployment to execution and recording of results. Once deployed onto the blockchain, the contract earns a unique address for identification. Authorized users can then activate the contract by initiating transactions containing the contract's address, which are executed by nodes or miners precisely following the contract's stipulations. The immutable nature of smart contracts ensures that once executed, a transaction's details remain unalterable, with access to resulting data controlled and restricted to authorized parties. This combination of transparency, automation, and security positions smart contracts as a transformative force across various sectors, reshaping the execution of agreements and management of business processes.

5.1.2 Consensus Algorithms

- Consensus Algorithms are essential in blockchain, determining which nodes have the authority to record transactions and facilitating their quick consensus

on the information to be included in a block. This guarantees the uniformity and safety of the data while simultaneously enhancing the computational efficiency of the blockchain [34]. In addition, these algorithms are essential to achieve Byzantine fault tolerance (BFT) (as described in D2.1 [32]) and preserve a consistent representation of the blockchain across all nodes in a decentralized network. The BFT mechanism aims to safeguard against system failures by employing a collaborative decision-making process that takes into account the input of both functioning and defective nodes, with the objective of minimizing the influence of defective nodes [34].

- **Proof of Work (PoW**): One popular BFT-tolerant consensus mechanism is the Pow consensus mechanism, where miners in Ethereum and Bitcoin must solve challenging cryptographic puzzles in order to validate transactions and add new blocks. The first miner who successfully solved the challenge is given permission to create a fresh block of transactions and append it to the blockchain. The solution to the challenge is verified by the other miners, and if it is correct, the new block is appended to their version of the blockchain. Nevertheless, this solution is not efficient in terms of energy consumption.
- **Proof of Stake (PoS)**: PoS BFT-tolerant consensus algorithm that is gradually becoming more popular because of its scalability and energy efficiency. In PoS the next block is selected based on the stake (i.e., amount of cryptocurrency held by the miner), instead of the computational power. The nodes that are responsible to validate the new blocks are chosen by staking a certain amount of cryptocurrency. The selection algorithm combines a mix of the candidate's stake (quantity of cryptocurrency possessed) and additional variables, such as coin age and randomization, for guaranteeing fairness across every node on the network. With the mining of new blocks, the energy of the spent coins diminishes slightly, leading to a deflationary mechanism where the total amount of currency gradually decreases, potentially raising its worth. Conversely, cryptocurrencies that experience an increase in quantity over time generally depreciate in value.
- **Delegated Proof of Stake (DPoS)**: allows token holders (stakers) to delegate their voting power to delegates or witnesses who are responsible for creating new blocks and validating transactions on the blockchain. Platforms like Tron and EOS employ the developing DPoS method, adding a democratic element to PoS.
- **Proof of Authority (PoA)**: is a new family of BFT consensus algorithms, designed to optimize the PoS mechanism while it is more focused on private and permissioned blockchains. It relies on a predetermined set of transaction validators based on their identity or reputation staked in the network. Current validators have the ability to vote for the inclusion of more users into the authority group [35].

Frequently employed in permissioned or private blockchains, BFT algorithms give priority to network speed and throughput over decentralization. The trade-offs between security, decentralization, scalability and energy consumption are unique to each consensus method. The blockchain network's unique needs and objectives, such as attaining maximum security, scalability, or energy efficiency, determine the consensus mechanism to be used. In order to overcome the shortcomings and restrictions of current methods, new consensus algorithms are being investigated and created as blockchain technology advances.

5.1.3 Access Control Mechanisms

Access control mechanisms are essential in blockchain networks, guaranteeing that only authorized users can engage with the system and carry out particular operations. These systems regulate the authorization levels given to users, determining their capacity to view, modify, or execute smart contracts and retrieve data stored on the blockchain. Various access control models and strategies are utilized to ensure the enforcement of security and privacy in blockchain ecosystems. Role-based Access Control (RBAC) and Attribute-based Access Control (ABAC) are the most popular ones.

Role-Based Access Control (RBAC) is a widely used access control method in blockchain technology that assigns permissions to individuals based on their roles within an organization or system. It involves defining specific roles like administrators, validators, and regular users, each with different access privileges. RBAC can also be implemented within blockchain networks to manage user permissions based on their roles.

Attribute-Based Access Control (ABAC) is a more flexible approach that considers multiple properties of people, resources, and the environment when determining access permissions. These attributes may include user roles, time of access, location, and other contextual information. It can be used in smart contracts on blockchain networks to enforce detailed access control policies.

5.1.4 Blockchain platforms

5.1.4.1 Hyperledger Fabric

Hyperledger Fabric is a permissioned blockchain framework designed for enterprise use, hosted by the Linux Foundation. It provides a modular architecture that allows for flexibility and scalability, allowing components such as consensus services to be plug-and-play based on the enterprise’s needs. Fabric supports smart contracts, known as "chaincode," and offers features like privacy through channels and private data collections. More specifically, Fabric allows the creation of different “channels” for transaction isolation, while offering the “private data” feature which enables the sharing of hashes as transaction evidence on the ledger. Although there is potential for the Fabric to support certain Ethereum 2.0 implementations, the new architecture

has not yet been fully implemented in the Ethereum public (i.e., permissionless) network, while Hyperledger Fabric has already achieved its version 2.0 milestone. Hence, such as support is yet to be offered [36].

5.1.4.2 Hyperledger Besu

Hyperledger Besu is an Ethereum client developed under the Hyperledger Foundation, providing enhanced flexibility and functionality. It supports both public and private Ethereum networks, as well as hybrid networks. Besu offers features like privacy through private transactions, permissioned network access[7]. It further supports flexibility in terms of selected consensus mechanisms like Proof of Authority (PoA) and Proof of Work (PoW)[8], as discussed in Section 5.1.2. One of the key features of Besu is its compatibility with the Ethereum Virtual Machine (EVM)[9]. This means that Besu can execute smart contracts and decentralized applications (DApps) written in Ethereum's native programming languages, such as Solidity, adhering to the Ethereum network's principles. Moreover, Besu is scalable and capable of supporting a large number of transactions per second.

5.1.4.3 Hyperledger Indy

Hyperledger Indy is distributed ledger solution focused on a decentralized identity management. It enables the creation, management, and verification of digital identities, offering features like privacy, interoperability, and self-sovereign identity (SSI) solutions, providing the necessary tools and libraries which conform to the Wide Web Consortium (W3C) standards [37]. Indy is particularly suited for applications requiring secure and privacy-preserving identity solutions, offering interoperable identities across different administrative domains and applications. Moreover, Indy promotes privacy-preservation through Zero Knowledge Proofs, which prove the trustworthiness of certain data in a collection of claims without disclosing any extra information, such as the identity of the individual providing the proof.

A comparative analysis of the aforementioned Hyperledger solutions is presented in Table 3.

Table 3 - Comparative analysis of different Hyperledgers

Criteria	Hyperledger Fabric	Hyperledger Besu	Hyperledger Indy
Consensus Mechanism	Organisation may choose (i.e., Pluggable)	PoW, PoS, PoA, etc.	N/A
Privacy Features	Channels, Private Data Collections	Private Transactions, Permissioning	Privacy, Self-Sovereign Identity

[7] https://www.hyperledger.org/blog/2019/08/29/announcing-hyperledger-besu
[8] https://besu.hyperledger.org/private-networks/how-to/configure/consensus
[9] https://www.hyperledger.org/projects/besu

Smart Contract Support	Yes	Yes	Limited
Smart Contract Language	Go, JavaScript, and Java	Solidity	N/A
Use Cases	Enterprise Applications,	Ethereum-based Applications	Decentralized Identity, Self-Sovereign Identity

5.1.5 PRIVATEER's Innovation in Blockchain

After thorough evaluation, Hyperledger Besu was selected as the preferable option for the PRIVATEER project due to its support in both private and public blockchain networks. Besu offers strong privacy features and a high capacity for processing a large volume of transactions, which is specifically interesting for PRIVATEER considering that trustworthiness evidence may contain a lot of information. In addition, the compatibility of PRIVATEER with Ethereum is highly useful as it enables the utilization of the wide range of tools and applications offered by the Ethereum ecosystem. This compatibility also applies to the integration with Town Crier oracles, which are specifically intended to support Ethereum, providing a secure mediator to the Blockchain network. The Town Crier oracles are further elaborated on Section 5.2.1.1.

Hyperledger Besu is the optimal blockchain platform for PRIVATEER's objectives of improving and facilitating the security and trustworthy data exchange, offering a protected and auditable distributed network that can be easily accessed by all interested parties. Furthermore, Hyperledger Besu's open-source nature and active development community makes it a secure, efficient, and flexible blockchain solution.

In addition to the Hyperledger solution, certain mechanisms as introduced in Chapter 2 and further elaborated on Section 5.2 are employed, enabling PRIVATEER to offer a secure framework which further allows privacy preservation. These mechanisms are summarised in Table 4.

Table 4 - PRIVATEER Blockchain mechanisms towards advanced security

Desired Characteristic	PRIVATEER component
Auditability	Hyperledger Besu which supports the integration and execution of smart contracts, which enable the mediating and management of trust-related information, exchanged via the ledger.
Integrity	Secure Oracle is integrated, providing integrity of data and processes. More specifically, the execution of data veracity checks over the received data (i.e., attestation reports from the Security Probe) are performed in a protected environment. Likewise, the smart contract creation is adheres to the same notion.

Access Control mechanisms for information in the ledger	Attribute Based Access Control (ABAC) mechanisms leveraging the Verifiable Presentations (VPs) are supported ensuring that only entities with appropriate attributes have access to the information available on the ledger.
Confidentiality in off-chain storage	Attribute Based Encryption (ABE) mechanisms employed for the protection of information (i.e., evidence) stored in the off-chain storage.
Privacy Preservation for external entities that access data in the ledger	Permissioned functionality of Besu is leveraged providing restricted access. In addition, Trust Exposure Layer is proposed by PRIVATEER, which provides the harmonisation mechanisms, ensuring that only information regarding the Level of Trust is exposed to external (to the MNO infrastructure entities). Details regarding the information used to perform the Level of Trust assessment (i.e., trustworthiness evidence, attestation report, etc.) is available only to internal entities.

5.2 Protocol description

5.2.1 Building Blocks

As described in the previous paragraphs, PRIVATEER leverages the Hyperledger BESU as the DLT solution. The next paragraphs analyse the internal building blocks and architecture of the selected DLT solution, further considering the overall scope of this framework in the context of PRIVATEER. Figure 6 provides an illustration of the Conceptual Architecture for the PRIVATEER Blockchain Framework, further clarifying the interactions between the components.

5.2.1.1 Secure Oracle – Town Crier (TC)

Blockchain oracles are external services that provide smart contracts with supplementary information. They serve as mediators that establish connections between blockchains and the **outside world**. Blockchains and smart contracts do not have the capability to access outside of the chain data, which refers to data that is located outside of the network. However, it is essential to get relevant information from external sources to achieve the operational goal of the blockchain, particularly in circumstances where smart contracts need such information to efficiently execute their business logic. In the context of PRIVATEER this functionality is critical to provide the attestation result and the attestation evidence, as acquired by the Security and µProbes (as described in Chapter 2).

Blockchain oracles further act as middlemen that **link off-chain and on-chain data** [38]. Oracles are essential components of the blockchain ecosystem since they enhance the capabilities of smart contracts by serving as an intermediate layer that acquires, validates, and verifies data from other sources, and then transfers that information.

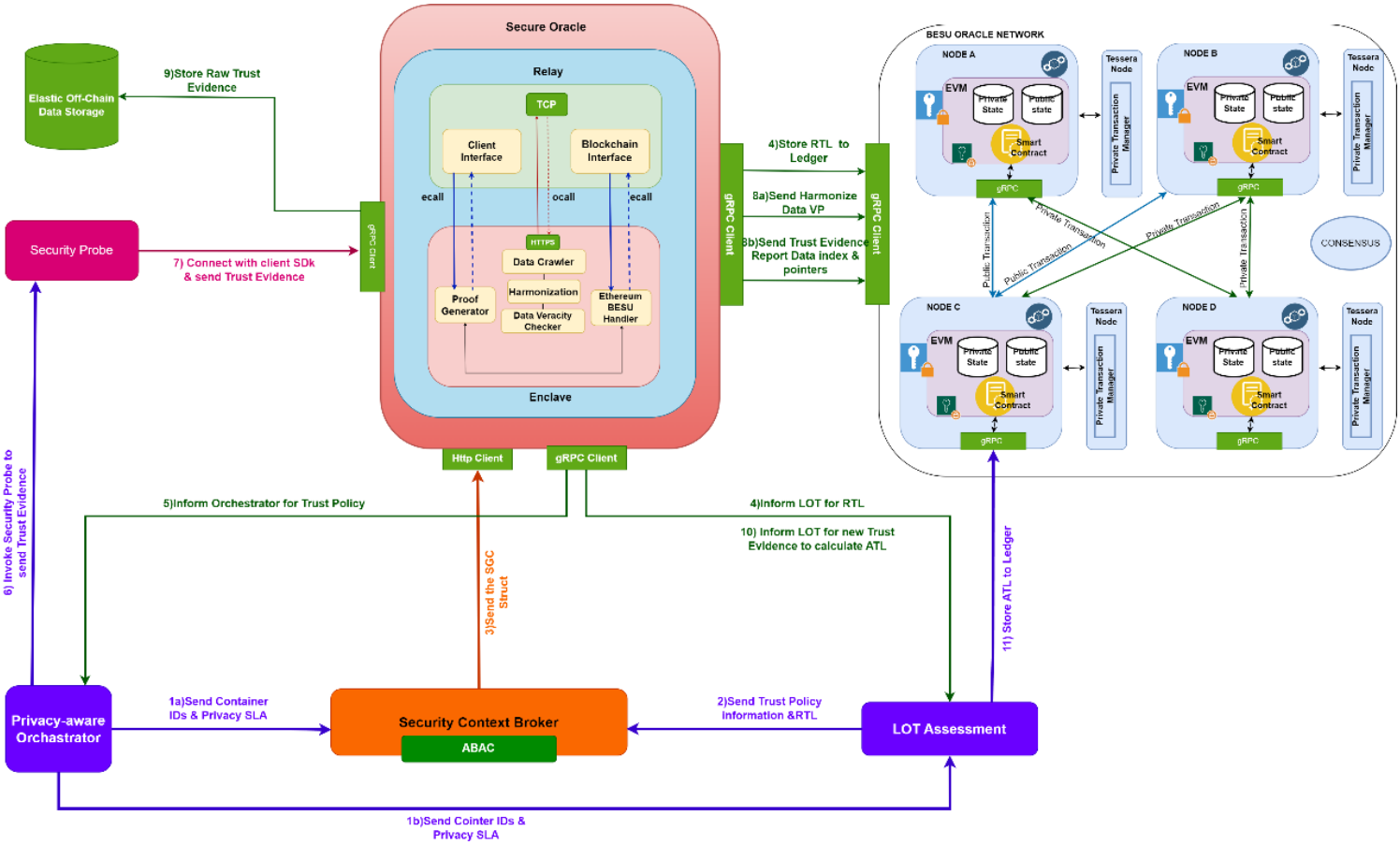

Figure 6 - PRIVATEER Blockchain Infrastructure Conceptual Architecture

The Oracles include qualities that PRIVATEER seeks to utilize and enhance for the purpose to achieve the project's vision of auditable trust evaluation tailored to the needs of the B5G landscape. Figure 6 illustrates the placement of the Secure Oracles network between the edge devices and the BESU network. This positioning allows for the gathering and filtering of data originating from the devices. Regarding data processing, the Oracles are utilized to receive attestation-related data in PRIVATEER. They then proceed to filter, standardize, and authenticate the data utilizing the Verifiable Presentations (VPs), as data structures, which may encompass both identity but also device state attributes. More information on the VPs is available in Chapter 6.

To access data from external sources, the smart contracts need to be triggered and network resources must be utilized. The Town Crier (TC) possesses the capability to not only transmit information to smart contracts, but also to send it back to other sources. Furthermore, TC combines a blockchain interface and a secure hardware component built on top of Intel SGX. It also leverages communication channels enabled with HTTPS to deliver source-verified data to smart contracts. The system facilitates confidential data requests by utilizing encrypted parameters, promoting secrecy. TC enables the secure utilization of credentials to access data from protected

data sources thanks to the implementation of Intel SGX. This feature will be utilized in PRIVATEER to allow Attribute-based Encryption (ABE), which will encrypt sensitive data before it is placed on the off-chain data storage (more information on the off-chain storage are available in section 5.2.1.7).

The PRIVATEER consortium also chose TC because of its **safe execution environment**, which is powered by Intel SGX. TEEs provide a secure setting that ensures the privacy and integrity of data and code, which is essential for oracles that deal with sensitive external data. Secure oracles built on Intel SGX's secure enclave technology are impervious to manipulation and unauthorised access, which boosts the reliability of the PRIVATEER blockchain. Data processed by the Town Crier oracle remains encrypted and confidential even while computations are being performed, thus in PRIVATEER data filtering and veracity operations are executed as part of the enclave to guarantee the isolated and confidential execution of sensitive data processing operations. In Figure 6 the enclave of the TC nodes is depicted, which includes the processes outlined before. Furthermore, the enclave's isolated environment guarantees the isolated execution of cryptographic operations, such as ABE of data, before they are stored on the BC infrastructure. Finally, the Town Crier may use the Proof Generator to cryptographically prove that the data processed and retrieved within the safe enclave has not been altered. This would enable blockchain apps to confirm that the oracle's outputs are valid.

5.2.1.2 Enclave

The Enclave, an execution environment supplied by Intel SGX, is a crucial component of the Town Crier (TC) design. It serves as an isolated location for data processing. The Enclave's primary responsibility is to process datagram requests from the blockchain by accessing external data sources that support HTTPS to get the required information (see Figure 7).

The Enclave places utmost importance on ensuring the secrecy and integrity of data processing. It functions with an awareness that it is completely isolated from potentially hostile operating systems and other activities on the host. The isolation is a fundamental aspect of the TC's security concept, allowing the Enclave to operate as a reliable component inside the larger system.

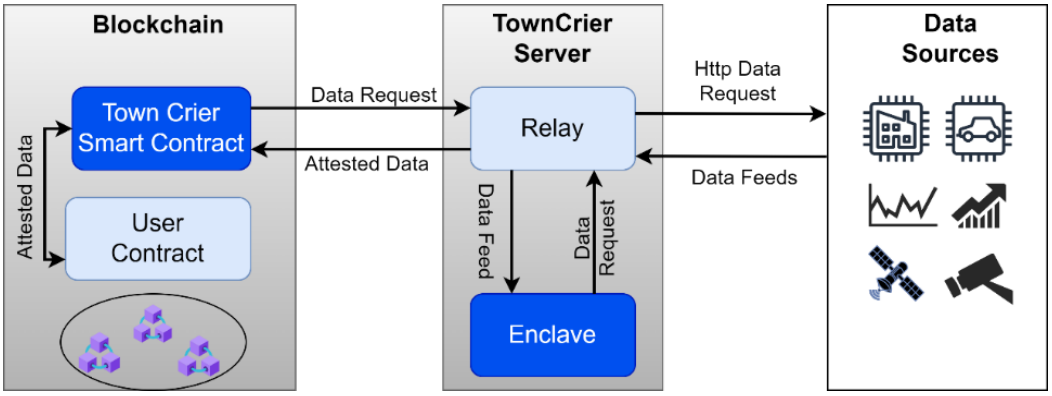

Figure 7 –System Architecture of TownCrier

5.2.1.3 Relay

The Relay serves as an infrastructure to facilitate communication **between the Enclave and the outside world**. Due to the constraints of SGX, the Enclave does not have direct network access. Therefore, the Relay plays a crucial role in managing bidirectional network traffic. This involves establishing a connection between the Enclave and the blockchain to monitor the status of the TC Contract and publish messages on the blockchain. It also involves managing off-chain service requests from clients for Enclave attestations, as well as facilitating communication between the Enclave and other data sources. The operation of the Relay is essential for the smooth transmission of information required for the functioning of the TC system.

Nevertheless, the Relay functions within the user space and does not receive any advantages from SGX's integrity safeguards. This vulnerability exposes it to exploitations by a hostile operating system, which might result in delays or failures. Despite this susceptibility, the design of TC strives to alleviate the consequences of such concerns. Although the Relay can potentially be used to carry out denial-of-service attacks, the TC design guarantees that it is incapable of generating inaccurate datagrams or causing customers to loss revenue paid for datagram services. TC's dedication to upholding the integrity and dependability of its service is emphasized by this design philosophy, which addresses the possible security problems presented by the Relay component.

5.2.1.4 Hyperledger Besu - PRIVATEER Private/Public Ledger

Hyperledger Besu's architecture offers a solid and flexible framework for developing enterprise-grade blockchain applications. As stated, Besu's characteristics are capable of satisfying the demands of PRIVATEER. Figure 6 presents that Besu is a structure constructed of crucial parts that provide its functionality.

- **Network Layer**: The BESU network of PRIVATEER will consist of numerous nodes. The nodes in the network engage in communication with each other to

achieve synchronization of the blockchain, dissemination of transactions, and active involvement in the consensus process. BESU is compatible with several networking protocols. PRIVATEER will utilize Ethereum's devp2p protocol, facilitating safe discovery and connection between nodes. After the nodes are linked, both public and private transactions can occur.

- **Consensus Mechanism:** BESU is capable of supporting many consensus mechanisms, such as Proof of Work (PoW) and Proof of Authority (PoA). The consensus mechanism governs the process by which transactions are authenticated and included into the blockchain. In the Proof of Work (PoW) consensus mechanism, miners engage in a competition to solve intricate mathematical problems to append new blocks to the blockchain. On the other hand, in the Proof of Authority (PoA) consensus mechanism, block validators are selected based on their established identity or reputation inside the network. The consensus method used for PRIVATEER is PoA.
- **Ethereum Virtual Machine (EVM)**: Besu incorporates a completely compatible implementation of the Ethereum Virtual Machine (EVM). The EVM is a confined runtime environment that plays a vital role in executing smart contracts and handling transactions in an Ethereum blockchain network. The EVM carries out bytecode, which is the compiled version of smart contract code, to guarantee the consistent execution of contracts across all nodes.
- **Permissioning and Privacy:** PRIVATEER utilized a hybrid technique to fulfil the data confidentiality criteria. Therefore, each node inside the blockchain network has the responsibility of keeping and overseeing the public global state, along with an individual private state for every privacy group. The private states include sensitive data that is not revealed in the globally replicated world state. In relation to data stored in an off-chain storage, the private state comprises the data hash and a reference pointer that indicates the exact location of the stored data off-chain. By employing this method, the private entity safeguards the essential information required to validate and authenticate the integrity of data. Access to the private state is provided by the incorporation of the private transaction manager within the Tessera node. The term "public state" refers to information that is freely accessible and may be used to exchange attestation-related data with trustworthy parties.
  This component is valuable in PRIVATEER as it provides transparency regarding the condition of an ecosystem. It may be utilized as a feature to facilitate external entities in conducting audits and certifications. Additionally, relevant information, such as source linkages that show the provenance of the private states from which the data came, is also retained. This setup guarantees that the necessary data needed for validation and verification may be acquired from the public ledger, while simultaneously safeguarding the privacy and security of the sensitive data stored off-chain.

- **APIs and Interfaces**: Besu provides a range of APIs and interfaces that allow users to communicate with the blockchain. These consist of JSON-RPC APIs for automated access, GraphQL APIs for retrieving blockchain data, and WebSocket APIs for receiving real-time updates. In addition, Besu offers command-line utilities and a web-based interface for monitoring and controlling the node.
- **Plugins and Extensions:** Besu's design facilitates expansion by including plugins and extensions. We leverage this property in PRIVATEER to enhance Besu's capabilities through the implementation of bespoke modules. This functionality enables us to incorporate further customized functionalities, such as integrating Elastic as the off-chain data storage for the project.
  The primary goal of Hyperledger Besu's design is to offer a secure, scalable, and extensible framework for developing blockchain applications for enterprises. It supports permissioning and privacy features, has a flexible design that works with Ethereum, and is suitable for many use cases across multiple industries.

5.2.1.5 Security Context Broker (SCB)

The Security Context Broker (SCB) serves as the sole middleman between the **blockchain infrastructure and external entities**. It operates as an intermediary, facilitating communication and interaction between the PRIVATEER components and Secure Oracle blockchain smart contracts, via APIs. The primary role of the Context Broker is to trigger the implementation of smart contract functionalities. It is a service that facilitates the transmission of messages and allows for the connection of different components to the Blockchain. It efficiently manages access control and permissions in the Distributed Ledger Technology (DLT) by leveraging smart contract technology. To guarantee and maintain safe access, Verifiable Presentations (VPs), which function as user tokens, undergo a process of verification. VPs not only grant access to a service, but also encompass specific attributes that define the user or device's privileges inside the network, as well as identity claims and proof of the system's secure status. VPs play a pivotal role also in terms of the Attribute-Based Access Control (ABAC) in PRIVATEER. The ABAC is implemented through the interaction between the Context Broker and the Peers in the Distributed Ledger Technology (DLT) network. This implementation takes use of the VPs, which also represent the security status of an object. Furthermore, the Context Broker possesses the potential to engage with external storage systems that are not integrated into the blockchain. Consequently, the SCB manages both communication and access control.

5.2.1.6 Attribute-Based Access Control (ABAC)

Attribute-Based Access Control (ABAC) is an advanced access control technique that uses characteristics as the basis for governing access to resources. ABAC does not

primarily rely on roles or rules, but instead assesses access requests by considering several factors connected with individuals, resources, and environmental variables. Examples of such features may encompass user roles, department, location, time of access, and the sensitivity of resources. The essential elements of ABAC consist of the Policy Decision Point (PDP), which assesses access requests based on predetermined rules, considering the characteristics of the person making the request, the resource being accessed, and the surrounding circumstances in order to make well-informed access determinations.

The Policy Enforcement Point (PEP) is responsible for implementing the access choices determined by the Policy Decision Point (PDP), therefore guaranteeing that only users with proper authorization are granted access to the requested resources. The Policy Information Point (PIP) obtains supplementary attribute information that is necessary for the Policy Decision Point (PDP), while the Policy Administration Point (PAP) oversees and establishes the policies, enabling administrators to create, modify, and remove policies according to the organization's needs. ABAC has several advantages, such as precise access control, the ability to react to changing situations, the capacity to handle complicated structures, and the flexibility to manage policies. ABAC offers enterprises a versatile, accurate, and dynamic method for access control, improving both security and operational efficiency. The PRIVATEER consortium chose to use ABAC as the mechanism for performing access control to the information stored on the BC due to its specific properties.

In PRIVATEER, any data sources (i.e., orchestrator) and external entities that require access to the BC infrastructure, either for data retrieval or data submission, must go via the trust-aware authorization and authentication process, which is based on ABAC. During operation, a device creates compliance certifications in the form of Verifiable Presentations. Therefore, these VPs represent the characteristics and protected condition of devices and entities. The PDP point in PRIVATEER is located within the Security Context Broker components, as seen in Figure 6. Every request is accompanied with the requisite VPs from the entities involved in the transaction in order to verify the identity, secure status, and permissions of the entity. The importance of the ABAC service supported within PRIVATEER SCB is seen in its role of managing and verifying user access to recorded blockchain data and off-chain data storage queries. Ensuring data security and exact access is extremely important in these scenarios, making this service indispensable.

5.2.1.7 Off-Chain Data Storage

The Off-Chain Storage feature of the PRIVATEER platform offers an innovative solution for managing a vast amount of (confidential) data. Such data that cannot be stored directly on the DLT network due to privacy and capacity limitations. Leveraging the off-chain storage is a common methodology in sophisticated blockchain applications, since it allows optimization of the performance, essential when managing the large

amounts of data. In this approach, the actual data is stored in Off-Chain Storage, which is separate from the blockchain network. Pointers are maintained in the private ledger of the DLT network to establish a connection between the data and the blockchain. These pointers function as references or links to the precise location of the data in the Off-Chain Storage. By using this approach, the system successfully overcomes the disparity between the blockchain's need for integrity and transparency and the practical limitations of storing big amounts of data on the blockchain. It functions primarily as an indexing service. In the context of PRIVATEER data stored in the off chain may include (attestation) evidence, policies, or system traces.

In order to enhance security and control access, the data stored off-chain is encrypted or encoded. The implementation of this encryption layer is essential as it effectively prevents unauthorized users from getting access to or altering off-chain data. Access to these pointers, and hence to off-chain data, is tightly controlled and limited to authorized users/entities with the proper privileges (according to ABAC). Users or entities have the ability to retrieve off-chain data by decrypting or decoding these data points.

The SCB exclusively regulates interactions with Off-Chain Storage in PRIVATEER. This broker functions as an intermediate, overseeing and facilitating all communication and data exchanges between the blockchain network and Off-Chain Storage. This design not only streamlines data retrieval and storage, but it also provides an additional layer of security by enabling the SCB to enforce various checks and limits to ensure that only authorized requests are executed.

The Elastic Search operates as the main storage solution for the PRIVATEER off-chain storage component, incorporating PRIVATEER-specific functionalities for handling off-chain data. Elastic offers storage capabilities for storing large amounts of data, using collections and documents instead of conventional tables and rows. Documents are composed of key-value pairs, whereas collections comprise both sets of documents and functions. A database is comprised of collections that include separate documents, each of which may have varied fields and sizes. Elastic's data structure optimizes the arrangement of hierarchical links and effectively stores arrays. The scalability of Elastic is a consequence of its advanced architecture, making it an excellent option for PRIVATEER's storage solution.

5.2.2 PRIVATEER Blockchain Infrastructure High-level Design and Flows mediated through Town Crier

As previously stated, PRIVATEER utilizes Blockchain technology to facilitate the exchange of data required for the Level of Trust calculation. The PRIVATEER Blockchain solution utilizes Town Crier as a Secure Oracle to offer enhanced security capabilities, and off-chain storage for safeguarding large volumes of data, such as attestation

evidence. Towards this direction and delving into the flow as firstly described in Chapter 2, the flow as it pertains to the functionalities that take place within the Blockchain is illustrated in Figure 8.

The flow is initiated when the Privacy-aware Orchestrator notifies simultaneously both the Security Context Broker (SCB) and the LoT Assessment component about the deployment of a new service. This notification further includes the Privacy SLA and the Service Graph Chain topology **(steps 1 and 2)**. The latter, the LoT Assessment component, leverages the received information to generate the Trust Policy for the specific service **(step 3)**, which is later transmitted to the SCB **(step 4)**.

The SCB leverages the received information to create the SGC Trust Chain Data Structure, as defined in Table 6, which further includes the Trust Policy structure for the designated service graph chain, as defined in Table 7, and the Privacy SLA **(step 5)**. These structures are elaborated in Section 5.2.3.1. After the creation of the SGC Trust Chain Data Structure, the SCB forwards this structure to the Town Crier **(step 6)**, which subsequently transmits it to the Oracle Contract **(step 7)**. The Oracle Contract establishes the SGCTrustChain() Smart Contract and may further compute the Real Trust Level (RTL). Please note that the calculation of the RTL functionality will be introduced in Release B of the platform. Subsequently, the Oracle Contract publishes the smart contract on the ledger leveraging the Hyperledger BESU Interface **(step 9)**. The Oracle Contract returns the Smart Contract which was previously published on

the ledger (containing the Trust Policy structure along with other information), to the SCB **(step 10)**, which then sends it to the Privacy-aware Orchestrator **(step 11)**.

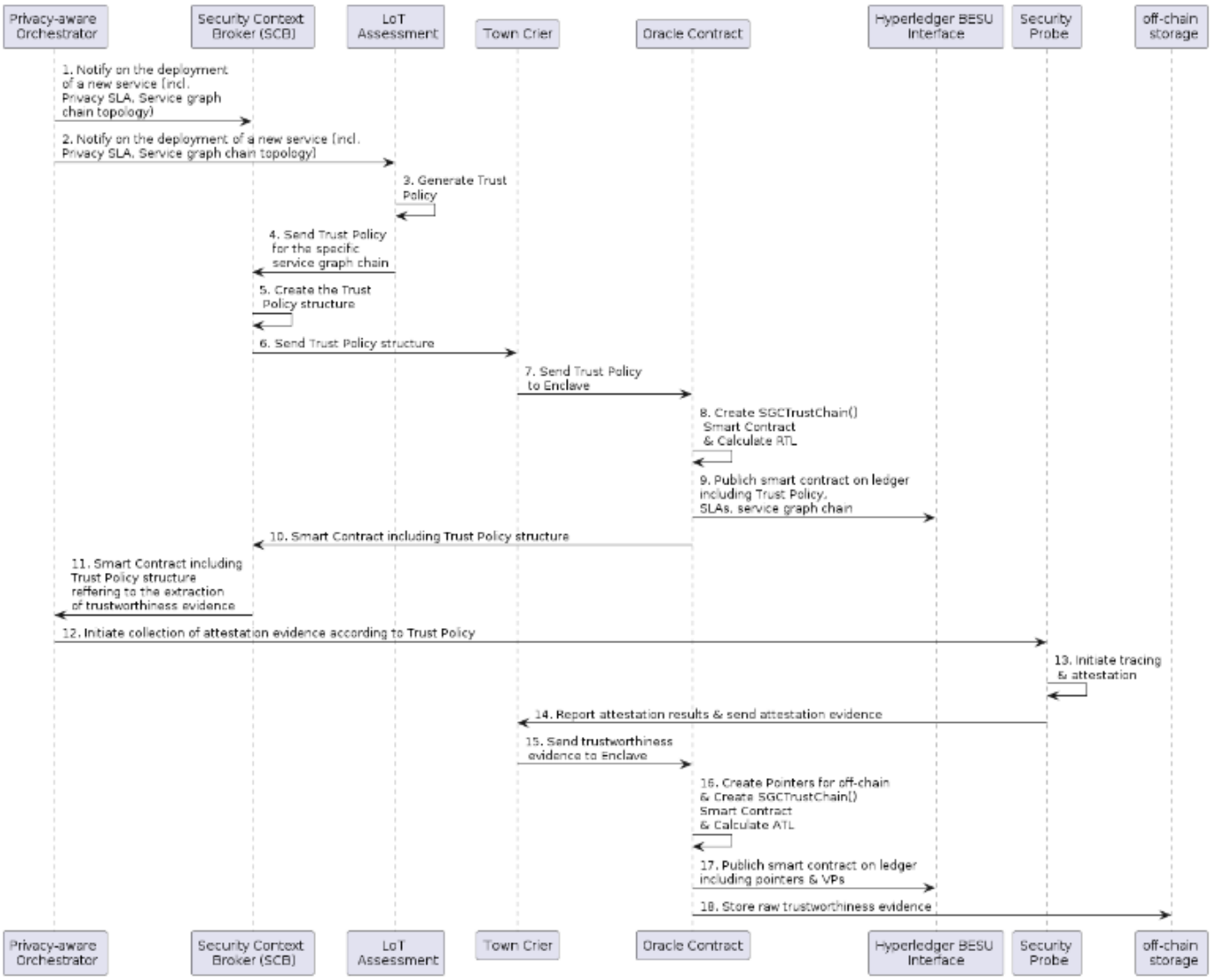

Figure 8 - Trustworthiness Evidence Management Mediated through Town Crier

Upon receiving this information from the SCB, the Privacy-aware Orchestrator begins the collection of the attestation report(s) and attestation evidence, based on the Trust Policy, prompting the Security Probe **(step 12)**. In essence, the Trust Policy defines whether attestation evidence is required for the LoT calculation for the specific graph chain, and the periodicity that this data should be collected, prompting the Security Probe to collect traces and perform the attestation task **(step 13)**. More information on the runtime attestation flow is available in Section 3.2.3.2. The Security Probe transmits the attestation report, along with the attestation evidence to the Town Crier **(step 14)**, which subsequently relays it to the Oracle Contract **(step 15)**. The latter, creates the pointers to the off-chain storage and constructs the Trustworthiness

Evidence Object Data Structure, as defined in Table 8 **(step 16).** Next, the Oracle Contract is deployed on the ledger using the Hyperledger BESU Interface **(step 17)** and the raw trustworthiness evidence are stored in the off-chain storage **(step 18)**. The LoT assessment can leverage the available information, through the Hyperledger BESU Interface, to calculate the runtime Actual Level of Trust (ATL) and updates the SGCTrustChain() Smart Contract accordingly. The ATL structure is defined in Table 9.

5.2.3 PRIVATEER Smart Contracts for Trustworthiness Evidence Management

In the PRIVATEER project, the implementation of Blockchain smart contract functions is driven by the objective to facilitate secure, transparent, and automated operations within the blockchain framework. The use of smart contracts in PRIVATEER introduces innovative approaches for managing trustworthiness evidence data like attestation evidence, CTI information and Proof of transit evidence of services and ensuring data integrity and confidentiality across the network. To this end, Table 5 details the smart contract functions that have been developed or are planned for future releases of the PRIVATEER platform.

Note that there are two types of smart contract functions:

- **Exported (E)**: Functions that are responsible to call and implement Smart Contracts (SCs) of the PRIVATEER BC, typically starting with an uppercase letter.
- **Unexported (U)**: Sub-functions and variables which necessitate a SC to operate as designed, usually starting with a lowercase letter.

Table 5 - Smart Contract Functions in PRIVATEER

Type	Smart Contract Function	Description	Release	Accountable
E	**GetTrustworthinessEvidence()**	It is the responsibility of this function to create a smart contract to govern the process of initiating and collecting the necessary of trustworthiness evidence such as (i.e., attestation evidence) from PRIVATEER Security Probe to Secure Oracle in accordance with the Trust Policy that applies. The Trust Policy is dictated to the Oracle from the LoT Assessment via Security Context Broker (SCB). Upon reveiving the data, the Secure Oracle executes the **validateDataVeracity()** and **StoreOfchainDataAndDBPointer()** function.	1.0	Oracle Function

U	**validateDataVeracity()**	Validates the integrity and authenticity of the trust evidence provided by **GetTrustworthinessEvidence()** based on the associated verifiable presentation that created by the Oracle when harmonizing the trust data. Integrity and authenticity of attestation results and evidence is performed through the verification of the associated signatures constructed from the underlying PRIVATEER Trusted Computing Base (instantiated in each node)	1.0	Oracle Function
U	**storeOfchainDataAndDBPointer()**	Appends metadata and a controlDBPointer to a previously trustworthiness evidence report that is signed from the Oracle enclave and stores it in the ledger. Raw traces are produced as part of the attestation evidence report, which are stored in an offline database. The pointer created with this function is stored on the ledger and points to these traces.	1.0	Security Context Broker
U	**harmonization()**	This function collects the trustworthiness evidence of every container from **GetTrustworthinessEvidence()** function, harmonize and create a verifiable presentation based on these trust evidence and stored on the ledger.	2.0	Security Context Broker
E	**SGCTrustChain()**	Its responsible to create a smart contract to collect the trust policy from the LoT Assessment component and related Privacy SLA and Service Graph chain topology from Orchestrator via SCB to calculate the **Requirement Trust Level (RTL)** of the service. At the second step the LoT Assessment component receives the trustworthiness evidence, including attestation reports (in the form of a VP) with the appropriate pointers to the off-chain storage where the actual attestation evidence may be acquired, if needed. The LoT Assessment component leverages the received information to calculate the **Actual Trust level (ATL)** of the service and compare it with the RTL to create a trust decision. The ATL is pushed to the ledger leveraging the SCB and the Secure Oracle.	2.0	Oracle Function
E	**QueryTrustLvl()**	This function is responsible to create a smart contract that implement the **trustExposure()** function in order to give access to the level of trust of a specific service to external entities/user.	2.0	Security Context Broker

U	**trustExposure( )**	This function verifies the credentials of the external entity or component that requests access, retrieves the entire history of the Trust Level of the particular service from the blockchain ledger and gives them access to the Level of Trust chain of the service if they are successful in passing the validation process.	2.0	Security Context Broker

5.2.3.1 Smart Contract Data Model Definition

Following that, we present detailed explanations of the data models employed within the Smart Contract framework. These models serve to support the operational requirements and functional specifications of PRIVATEER in relation to the delivery of the Trust Policy and enable the collection and execution of calculations based on several types of trust evidence. These models are represented as data structures, consisting of fields in the form of a JSON schema, which specifies the features that must be included in each structure.

Various data structures are defined based on the specific work environment and the data they are required to hold.

Table 6 - SGC Trust Chain Data Structure

Name	Data Type	JSON Schema	Description
PrivacySLA	string	json:" privacysla"	The Service Level Agreement (SLA) produced by Orchestrator's SLA Manager contains details on security, trust, privacy characteristics, as well as network and service capabilities. PrivacySLA indicates the users or external entities who are authorized to access specific containers and services.
ContainerID	[] integer	json:" servicegraphchain"	The array list contains the container IDs that make up the service graph chain, representing the services that possess these trust evidence in order Trust Level Manager performs the Trust Assessment.
TrustPolicy	[] struct	json:" trustPolicy"	The Trust Policy covers several characteristics that are essential for evaluating the trustworthiness of the service. These parameters include the trust level for the specific service, Proof of Transit (PoT), information using by the Oracle to collect trust evidence. In addition to the parameters, the Trust Policy contains details on the frequency at which the data should be queried (i.e., periodicity) and the trust relationships that are established based on the service graph chain.

TypeofEvidence	string	json:" typeofEvidence"	What type of evidence need to ask from Security Probe to collect on Town Crier (i.e., evidence for proof of transit and for attestation)
RequiredTrustLevel (RTL)	[] struct	json:" requiredtrustlevel"	The Required Trust Level (RTL) is a measure of the trustworthiness of the service graph chain prior to uploading the trustworthiness evidence to the blockchain, to perform the runtime assessment. It serves as a threshold that decides the level of security for a specific service.
ListOfAttributes	[] string	json:" listofattributes"	This field specifies the list of attributes that a given device or entity must possess in order to access and edit this specific data structure from the DLT.
Signature	[] string	json:" signature"	Include a self-signed certificate issued by the enclave (i.e., Secure Oracle). This certificate is applied at the hash of Smart contract. The enclave signature includes information that enables the Intel SGX architecture to identify whether any part of the enclave file has been altered.

Table 7 - Trust Policy Data Structure

Name	Data Type	JSON Schema	Description
ActualTrustLvl	[] struct	json:" actualTrustLvl"	The Actual Trust Level (ATL) is derived from the computation based on the trustworthiness evidence of services in order to be determined if the container of services meet the requirements of the RTL.
TrustModel	[] string	json:" trustmodel"	An array of strings is utilized to indicate the relationship between several design models from separate developers, based on their ID definitions, without disclosing the actual code that implements by them.
ListOfTrustSource	[] string	json:" listoftrustsource"	The objective is to monitor specific trust parameters (such as configuration integrity through attestation) and measure the Level of Assurance (LoA) of the associated service graph chain.
Periodicity	integer	json:" periodicity"	Periodicity refers to the frequency at which Oracle decide to request and collect trustworthiness evidence.

Table 8 - Trustworthiness Evidence Object Data Structure

Name	Data Type	JSON Schema	Description
DbPointer	[] string	json:" dbPointer"	A string identifier, that is used by the Oracle to store the value of the trust evidence after the data veracity check, in an off-chain database.
OracleSign	[] string	json:" oracleSign"	The unique signature that produced from the enclave SGX of Secure Oracle that apply to trust evidence before them store at the off-chain storage.
ContainerSign	[] string	json:" containerSign"	The signature is created and applied to the container from which the specified trust evidence originates.
CategoryTrust Source	[] string	json:" categoryTrustSource"	It refers to the type of trust parameter that the evidence concerns (such as, evidence for attestation and/or proof of transit)

Table 9 - Actual Trust Level Data Structure

Name	Data Type	JSON Schema	Description
TrustLvlValue	[] string	json:" trustLvlValue "	Indicate the Trust Level value derived from the calculation procedure to apply on Service Graph chain by LoT Assessment via the **SGCTrustChain**() smart contract using the trustworthiness evidence.
LotSign	[] string	json:" lotSign "	The string identification represents the Level of Trust Assessment, which is responsible for updating the new trust values each time new trustworthiness evidence collecting to Blockchain.
DbPointer	[] string	json:" dbPointer"	A string identifier, that is used by the Oracle to store the value of the trust evidence after the data veracity check, in an off-chain database.

5.3 Plan for development

The implementation roadmap for the Blockchain development is characterized by two major milestones. In the initial release, the primary objective is to establish core functionalities and provide a stable base for the PRIVATEER framework to operate. This mainly involves the construction and management of the necessary contracts to support the PRIVATEER workflows as presented in Section 2. This implies that the first version contains the full instantiation of the Town Crier secure oracle, along with the

Service Graph (SGC) smart contract. In addition to that the necessary gRPC and http clients are envisioned so as to enable the data sharing from the various PRIVATEER components towards the DLT. Hence, it enables the privacy-aware orchestrator to publish information pertaining to the SLAs and the service graph chain as well as the LoT Assessment to specify the necessary trust policies for each service. On top of that, communication between the Security Probe and the Secure Oracle is also envisioned so that the trustworthiness evidence is recorded to the DLT. Finally, the veracity of the ingress data to the Town Crier is also planned for the first release.

Building upon the foundation established in the first version, the second release introduces significant enhancements and additional features to regulate the access to the data stored in the Blockchain. In this second version, the SCB is enhanced with the full-fledged ABAC and ABE implementations so as to control the access to the information stored to the DLT. In addition, the final integration with the distributed identity management framework is envisioned. Last but not least, this second release accommodates the harmonization mechanisms (e.g., harmonization functionalities within Town Crier) required for the trust exposure layer to provide aggregated trust results to external actors.

6 Distributed Identity Management

The rapid adoption of the Internet of Things and 5G network further increases the need for secure communication among devices. Therefore, it is necessary for each device to have a unique digital identity in order to ensure secure communication and authentication [39] [40]. A plethora of the existing models for identity management leverage centralized identity providers and repositories to meet the goal of identification. However, this centralised identity management approach creates a single point of failure, rendering it vulnerable to widespread system failures and an attractive target for malicious individuals. This increases the risk of large-scale data breaches, potentially exposing significant quantities of personal information Moreover, centralized solutions frequently fail to meet strict privacy requirements, thus increasing the chances of unintentional data exposure [41].

By distributing control over personal data and leveraging secure, immutable ledgers, Self-Sovereign Identity (SSI) frameworks mitigate the risks associated with single points of failure and data privacy breaches, providing data subjects with greater autonomy and security over their digital identities [39] [42]. In addition, the principles of transparency and data minimization are also applied, as individuals are made aware of the purposes of data processing, and they only share necessary information depending on the specific purpose [43]. Since the majority of users own at least one digital identity, in order to authenticate themselves at a service or an application, the need for protecting these identities has grown exponentially [39]. Moreover, it is crucial to acknowledge that identities extend beyond individual users to further include services, which highlights the significance of strong mechanisms for protecting identities.

The following sections delve into the State of the Art (SotA), detailing on existing protocols, while introducing the PRIVATEER protocol descriptions, and the plan for development.

6.1 State Of The Art

In Self-Sovereign Identity (SSI) model, individuals have control over their own personal information without relying on centralized authorities. More specifically, users can securely manage and share their personal data using cryptographic techniques, ensuring security and privacy, when interacting with other entities [44]. Within an SSI ecosystem, individuals can store their identity information and share it with other parties when the occasion arises. In addition, they can also decide what information they would like to share with other entities, and they are not obliged to share data which is not required for the identification [43].

Self-Sovereign Identity is founded on decentralized and distributed ledger technologies. Along with DIDs and VCs these technologies (which are elaborated in sections 6.1.1 and 6.1.2 respectively) can be used in order to **create irrefutable and immutable (tamper-proof) records** [45]. The **privacy-by-design** principle is also upheld within the context of SSI [42]. More specifically, the data subjects can control when they will share their attributes, and which attributes (which data) they would like to share. With **selective disclosure**, data subjects have control over the data which is shared for a specific purpose. In addition, **no personal data is stored on the ledger**, neither encrypted nor hashed. This eliminates the need of monitoring or removing data from the blockchain either for privacy or security-related purposes [42].

The stakeholders involved within an SSI ecosystem are the holder, the issuer, and the verifier. Each entity plays a different role in Self-Sovereign Identity model [44] [46].

- Holder: An entity which has ownership and control over a set of personal information. Each holder can have one or more digital identities, without depending on a third party to obtain them, and can manage and selectively share their personal data using verifiable credentials [47] [48]. The holder stores these credentials in his/her digital wallet, which may be a software application or hardware device. By retaining control over their identity information, holders can assert their identity and share relevant credentials with verifiers, thereby preserving privacy and security in digital transactions [43].
- Issuer: A trusted entity which issues verifiable credentials on behalf of the holder. Issuers are responsible for verifying the authenticity of the information they issue and cryptographically signing the credentials to ensure their integrity [43]. By issuing verifiable credentials, issuers enable holders to present proof of their identity in a secure and verifiable manner.
- Verifier: An entity that verifies the authenticity of a verifiable credential provided by a holder. Verifiers validate the cryptographic signatures of the presented credentials to ensure their authenticity [47].

In terms of PRIVATEER architecture, as described in Chapter 2 of the present deliverable, the holder is the entity possessing the credentials (i.e., the MNO, the Service Provider or the user), the issuer can be any trusted entity issuing the DIDs (the VCs are self-issued in PRIVATEER), while the Verifier resides within the Security Context Broker component.

There are certain use cases where a regular user may leverage Self-Sovereign Identity. In general, SSI finds applications in Internet of Things (IoT) devices, payment solutions, public transportation, the healthcare sector, digital driving license verification [39] [43] [49] [50], and various other domains. As already mentioned in the deliverable D2.2 [2], a possible scenario for users leveraging DIDs is a city that leases a multi-domain B5G network to support various public and private transportation operators.

In that case, the Service Provider (i.e., which refers to the Ticketing System and Transport Provider), which reflects in essence an application deployed in an MNOs A infrastructure, is responsible for processing data regarding journey planning, routing, and fare settlement in a privacy-preserving manner. In this specific case, PRIVATEER aims to leverage Decentralized Identifiers and Verifiable Credentials to securely authenticate users (i.e., travellers) so that they can access transportation services even when moving across different regions served by different MNOs. In this scenario, the traveller is the holder of the Verifiable Credentials, the Identity Provider is the Issuer, whereas the Transport Provider is the Verifier.

6.1.1 Decentralized Identifiers (DIDs)

Decentralized Identifiers (DIDs) are considered as the cornerstone of SSI ecosystems, by offering a way to uniquely identify individuals, organizations, devices, and entities in a decentralized manner [42] [47]. Unlike traditional identifiers, like usernames or email addresses, DIDs are not related to any centralized registry or authority. On the contrary, they are cryptographically generated and stored on distributed ledgers, such as blockchains [43] [46]. Each DID is globally unique and user-centric, enabling individuals to assert ownership and control over their digital identities. DIDs can be associated with cryptographic keys, allowing users to authenticate themselves without relying on any intermediate entity to do so [43]. This decentralized approach increases the level of both security and privacy, as users maintain full control over their identity information. In essence, a DID is a Uniform Resource Identifier (URI) which refers to a DID subject and associates the DID subject with its corresponding DID Document [47] [51]. The DID document contains information, such as verification methods and cryptographic public keys, relevant to the DID subject [51]. According to the World Wide Web Consortium (W3C), DIDs are defined as “globally unique persistent identifiers” [51]. The specifications outline a method for verifying the public keys included in DID documents.

6.1.2 Verifiable Credentials (VCs) and Verifiable Presentations (VPs)

Verifiable Credentials (VCs) are digital documents, and more specifically digital credentials, which contain information about specific attributes or claims of a data subject. These claims are basically personal information of an individual, such as his/her age or level of education [44] [46]. Unlike customary credentials, which are issued by centralized authorities, verifiable credentials are designed to be independent and cryptographically verifiable. They are issued by trusted parties, known as issuers, and are cryptographically signed to ensure their integrity and authenticity [44] [48]. Verifiable credentials enable individuals to present proof of their attributes in a privacy-preserving manner, without disclosing any unnecessary and additional personal information [48]. The VC data model takes advantage of verifiable credentials in order to establish trust among all involved entities within the SSI ecosystem. By leveraging verifiable credentials, users can exchange their

information seamlessly in a decentralized and privacy-enhancing manner, across different platforms and services [43].

A Verifiable Presentation (VP) is a cryptographic proof that attests to the validity of the information being shared, without revealing unnecessary details about the underlying credentials or the individual's identity [41] [49]. This proof is generated by combining verifiable credentials with cryptographic signatures and proofs, ensuring the integrity and authenticity of the presented information [41] [43] [47]. Verifiable presentations allow individuals to selectively disclose specific attributes or claims from their credentials, depending on the context or the needs of the recipient.

6.1.3 PRIVATEER's Innovation in Identity Management

PRIVATEER pioneers in adopting SSI for Identity Management in the context of B5G networks, not only for user but also for infrastructure component verification. By adopting this approach, PRIVATEER leverages SSI structures, such as Verifiable Credentials (VCs) and Verifiable Presentations (VPs), to enhance network security while also taking into account issues related to privacy. Towards this end, access to specific information such as trust levels or the acquired trustworthiness evidence is granted though selective disclosure of attributes (i.e., ABAC).

In terms of employed DLT technology for Identity Management, PRIVATEER leverages the **Hyperledger Aries** [52], that is part of the extensive Hyperledger ecosystem, providing an **open-source** initiative designed to offer developers with a robust set of libraries and tools, facilitating the creation of Decentralized Identity applications [39]. This set of libraries and tools comprises features which are essential for the secure management and presentation of DIDs and VCs, while safeguarding user's privacy. Hyperledger Aries Cloud Agent Python (ACA-Py) [53] is part of Aries Hyperledger framework. ACA-Py implementation simplifies the integration of DID functionalities, and empowers developers to exploit its capabilities, within their Python-based projects. In addition to the toolset provided by Aries, **Hyperledger Indy** [10] further provides a Distributed Ledger Technology (DLT) specifically designed for DID management. Within Indy Hyperledger, developers are able to create and manage digital identities in a secure, privacy-preserving, and interoperable manner.

In the context of PRIVATEER, Aries Cloud Agent is preferred due to its robust implementation, active development community, and seamless integration capabilities [39] [43] [46]. ACA-Py offers a comprehensive suite of tools for building SSI solutions. It supports all functionalities needed in PRIVATEER, such as VC issuance, VC verification, VC revocation, and ensures flexibility and scalability, making it adaptable to diverse use cases. Moreover, ACA-Py benefits from extensive community support, providing regular updates and security enhancements.

6.2 Protocol description

PRIVATEER considers the entities as also defined by the W3C standards: i) the holder, ii) the issuer, and iii) the verifier. The holder and the verifier each possess a DID which is stored locally in their wallet, while the issuer's DID is published on the blockchain for Identity Management. Considering the PRIVATEER architecture, the holder may be any entity (i.e., covering both components and users), while the Security Context Broker (SCB) acts as the credential verifier. The main function of the issuer is to issue and revoke credentials on behalf of the holder, whereas the verifier is responsible for validating the holder's credentials. The scenario description is depicted in the figure below (Figure 9). It is worth noting that the blockchain used for identity management is based on Indy Hyperledger [10]. The flow goes as follows:

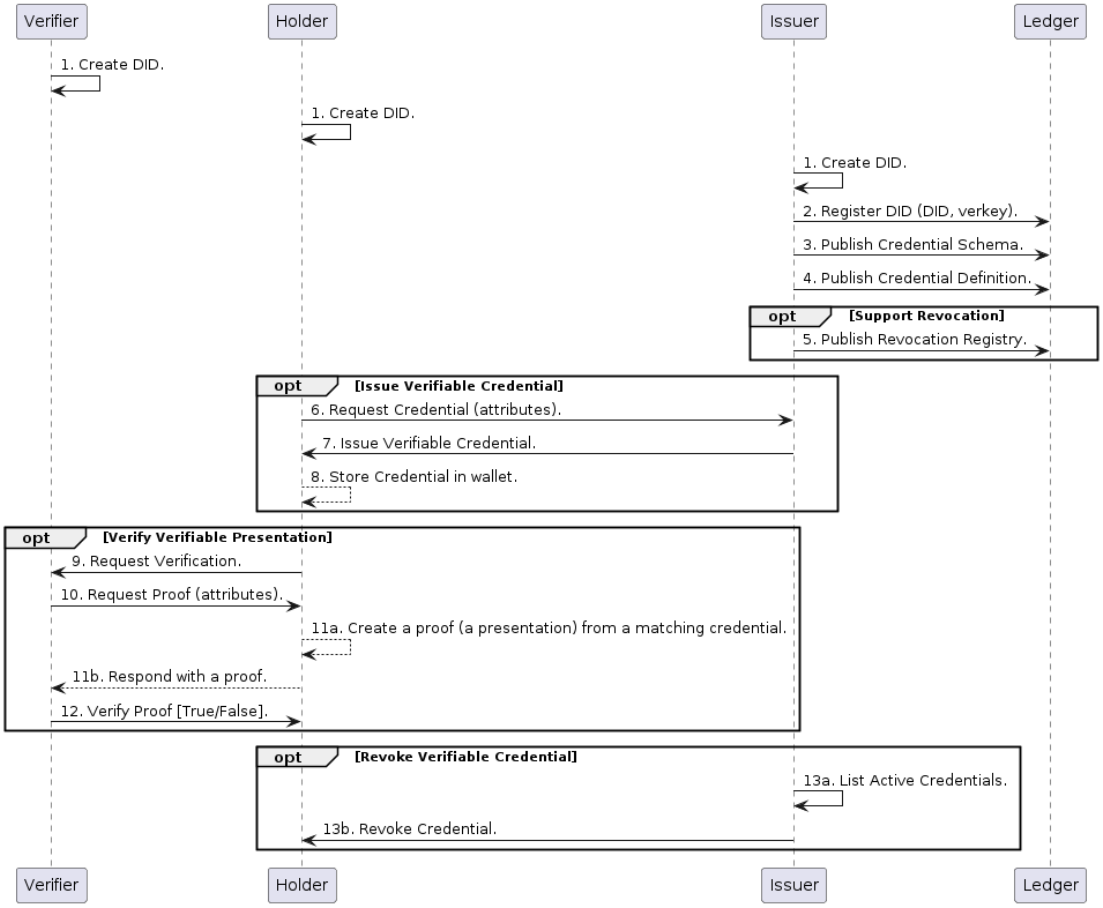

Figure 9 - Self-Sovereign Identity Diagram and Flow

The holder, the issuer and the verifier create a local DID in their wallet respectively **(step 1)**. In parallel, the issuer publishes their previously generated DID on Indy Hyperledger **(step 2)**. Along with the generation and publishment of the DID, the issuer creates and publishes a credential schema on Indy Hyperledger **(step 3)**. The credential schema consists of a name, version, and a set of attributes. More

specifically, the credential schema can be considered as a template for the upcoming credentials, and it defines the structure and the type of the attributes.

The issuer creates and publishes a credential definition on Indy Hyperledger **(step 4)**. The credential definition is an instance of the credential schema and includes cryptographic material, such as public keys, which are used for the issuance of verifiable credentials. If revocation of verifiable credentials is supported, the issuer should declare this information when registering the credential definition on the ledger. In addition, it should also define the size of the revocation registry, which determines the maximum number of credentials that can utilize this revocation registry. In case that revocation is supported, the issuer should also create and publish a Revocation Registry (RR) on Indy Hyperledger **(step 5)**. This is done automatically. In fact, two revocation registries are created, one active and one initiated. The advantage of this implementation lies in the absence of delays when the first registry reaches capacity, as the second one can be utilized. During continuous operations, when one revocation registry reaches its limit, the second revocation registry gets activated, and a new one is generated to ensure that one registry remains on standby.

The holder requests a verifiable credential from the issuer **(step 6)**. More specifically, the holder makes a credential proposal to the issuer, inquiring the latter to issue a credential based on specific attributes. The issuer inspects the credential attributes. If all attributes are valid and the holder meets the criteria, the issuer issues and sends the verifiable credential to the holder; otherwise, the issuer sends a message to the holder informing the latter that an error occurred **(step 7)**. The holder receives and stores the verifiable credential in its wallet **(step 8)**.

Whenever the holder wants to get verified somewhere, for instance, at a service, it must first establish a connection with the verifier **(step 9)**. The verifier requests a presentation (proof) from the holder **(step 10)**. The holder creates and responds to the verifier with a proof from a matching credential **(step 11)**. The verifier verifies the holder's presentation, yielding a Boolean result (true/false) **(step 12)**. If the attribute values are valid and the credential is active and not revoked, the result will be true; otherwise, it will be false.

The issuer can also revoke the previously issued verifiable credentials **(step 13)**. More specifically, the issuer can access all issued credentials at any time and select which ones to revoke. After credential revocation, the status of the credential will change, and the credential will not be considered as valid anymore. As a result, the holder will not be able to verify itself.

Note: It is worth noting that a connection must be established between the holder and the issuer before credential issuance, and between the holder and the verifier before proof verification.

6.3 Plan for development

The main functionalities which will be developed in the context of PRIVATEER are as follows:

1) DID Generation: A unique Decentralized Identifier, either private or public, is created for each entity within the SSI ecosystem.
2) Verifiable Credential Issuance: A Verifiable Credential is issued by an issuer and certifies various attributes or claims of a holder.
3) Verifiable Credential Validation: The verifier confirms both the authenticity and the integrity of the received verifiable credentials or presentations.
4) Verifiable Credential Revocation: The issuer, who has previously issued a verifiable credential on behalf of the holder, can revoke the holder's credential if needed.
5) DID Resolution: A DID is resolved into a DID document containing metadata or information associated with the entity it represents. When a DID needs to be resolved, the resolver locates the corresponding DID document, typically stored on a decentralized ledger, and retrieves relevant information such as public keys, service endpoints, or authentication mechanisms associated with this DID [51] [54]. This process enables verifiers to authenticate and interact with the DID subject in a secure, privacy-preserving, and decentralized manner.

For Release A (M16), a first version of all the above functionalities has been implemented. However, the current implementation needs to be improved in order for the DID/VC functionalities to be performed properly. In addition, the attributes which will be included within the verifiable credentials need to be defined and described since, at the time, only credentials with test attributes are issued. It should also be clarified how the SSI concept will adapt to the Use Case "Verification of Mass Transportation Application" and with which components it is expected to interact. Till Release B (M30) all open points should be addressed, in conjunction with other relevant partners, and an improved version is expected to be implemented.

7 Privacy-preserving CTI sharing

Sharing Cyber Threat Intelligence (CTI) information is crucial for improving the security of networks, including B5G networks. By facilitating the exchange of relevant data among interested parties, operators and other stakeholders, it is possible to efficiently identify and address imminent cybersecurity threats. However, organizations may hesitate to share this kind of data due to the potential exposure of sensitive information, such as internal network topology and configuration details. Disclosing such information could potentially provide malicious actors with valuable insights into an organization's infrastructure, thereby allow them to initiate focused cyber-attacks, resulting in data breaches, service disruptions as well as financial and reputation damage.

Therefore, although the exchange of cyber threat intelligence is crucial for improving network security, organizations must thoroughly evaluate the privacy consequences and implement suitable measures to reduce the risks associated with sharing confidential data. It is clear that enabling the **seamless** and **privacy-preserving** exchange of threat data is essential for protecting critical infrastructures. Usually, when users wish to retrieve information, they need to formulate a search query and send it to a server operated by a third party. The server then provides data that is related to the search query. Normally, this data is decrypted first, which allows for operations to be performed in clear text. This can bring up the confidentiality of the data exchanges into question. An effective method to address this issue is through **Searchable Encryption**.

Searchable Encryption is a technique that uses trapdoors to search and retrieve information from an untrusted third-party server without the need to decrypt. For example, a user that wants to search for information about “apples” will first create a trapdoor (the result of a secure one-way cryptographic function, eg. HMAC) from the keyword, which is then sent to the server. The server when receiving the trapdoor compares it with the contents of their database where the identifier of the record is the trapdoor. This requires that the encrypted inverse index is prepared and set up before searching is possible. Since all data is encrypted (even the search query), **confidentiality** is guaranteed. Searchable encryption can be split into different types based on the used cryptographic techniques, with the base ones, from which many other approaches are derived, being i) the Symmetric Searchable Encryption (SSE) and ii) Public Key Encryption Keyword Search (PEKS).

Symmetric Searchable Encryption

SSE uses symmetric key cryptography, allowing for anyone with access to the secret key to encrypt and decrypt data. SSE can either be static or dynamic. Static means that the encrypted index is set up and then cannot be updated without rebuilding the

encrypted index. Dynamic, on the other hand, allows for new records to be inserted into the index as well as existent records being able to be deleted. Usually, static SSE requires a combination of four steps:

- **Key Generation**: Generates a secret key. It usually receives a parameter, such as key size of the system, as input.
- **Set Up**: Generation and preparation of the encrypted index – receives the secret key and the documents that will be encrypted and outputs the index.
- **Trapdoor Generation**: Generation of a trapdoor for a certain keyword – receives the keyword and the secret key and outputs the trapdoor of the keyword.
- **Search**: Searches for all documents that have a certain trapdoor – receives the index and the trapdoor as input and returns a set of documents.

Dynamic Searchable Encryption

For dynamic SSE, the same four steps are used, along with an extra four:

- **Insert Token**: Generates a token used to call for an insert action - receives the secret key and the document that will be added to output an insert token.
- **Insert**: Adds the new document to the chosen encrypted index – receives the insert token and the encrypted index, outputting an updated index, now with the new document.
- **Delete Token**: Generates a token used to call for a deletion action - receives the secret key and the document that will be deleted to output a deletion token.
- **Delete**: Removes the chosen document from the chosen encrypted index – receives the deletion token and the encrypted index, outputting an updated index, now with the chosen document removed.

Public Key Encryption Keyword Search

PEKS uses public-key cryptography, allowing for only the owner of the corresponding private key to perform searches. This approach is preferred for use in multi-user scenarios where nonrepudiation is a requirement. A PEKS approach usually consists of four steps, analogous to the ones of the SSE approach:

- **Key Generation**: Generates a private/public key pair.
- **Set Up**: Generation and preparation of the encrypted index – receives the public key of the recipient and the documents that will be encrypted and outputs the index.
- **Trapdoor Generation**: Generation of a trapdoor for a certain keyword – receives the keyword and the private key of the recipient and outputs the trapdoor of the keyword.
- **Test**: Verifies if an encrypted document was encrypted with a certain token or not, returning a Boolean variable.

The following sections list some State of the Art (SotA) works, detailing on existing protocols, while introducing the PRIVATEER protocol descriptions, and the plan for development.

7.1 State Of The Art

7.1.1 Searchable Encryption

Searchable encryption was first proposed by Song et al. [55], who presented algorithms allowing for search over encrypted text data. This scheme uses a **deterministic encryption** scheme to encrypt the keywords, in a first round of encryption. Then it uses a stream cipher for a second round of encryption. For instance, in this scheme, a keyword is first run through this algorithm and the result is separated in two parts – one which will be used for key generation for a hash function and the other which will be XORed with a random seed, picked by the keystream, and the result of the hash function (which is computing the key generated by the first part along with the random seed). If a user wants to perform a search, they first encrypt the keyword, and this result is sent to the server which iterates over all encrypted keywords, attempting to recover the seed that was used in the second layer of encryption, by performing the XOR operation. After that, the key generated from the first part of the encryption result is compared with the cipher text. If it is a match, then the keyword that was being searched for was found. This approach has its' issues. For one, it doesn't achieve very strong security. While the ciphertext itself is secure, no security is achieved regarding search capabilities. The scheme leaks the position of the keyword in the document, which can lead to statistical analysis attacks. This scheme's search time is also linear, which means, the time required for a search to complete would increase, at an equal rate, depending on the number of keywords the documents are hosting.

Goh [56] then attempted to increase the security of the previous scheme through a **forward index approach**. In it, for each document there is a combination of keywords, which have been encrypted and linked to it. A user that has the secret key can generate a trapdoor and then search for a certain keyword. This scheme requires that the index be built beforehand – and this is done using bloom filters. These are data structures, in the form of an array of bits, each bit representing the presence of specific data. These are used to definitely confirm that an element is not part of a data set. All bloom filters will be empty when first created with all its bits set to zero. Once a new element is added, the bloom filter's array of bit changes, the bit that maps to the new element will become 1. Thus, by hashing the element the user wants to find, and checking whether all the positions returned of said bits are set to one, the user can verify that the keyword exists, in theory. However, this is not certain. Bloom filters will never produce false negatives, but they can, especially on larger datasets and depending on the number of bits available for each element, produce false positives.

Another issue is that the number of ones can be enough to give a clue of how many keywords the document is hosting. Once again, the issue of linear search time occurs in this scheme as well, since the search is performed by using the combination of encrypted keywords linked to the document.

Curtmola et al. [57] presented an **inverted index approach**. This approach would become the basis for many other searchable encryption implementations. The idea behind this approach is that, instead of trying to find a keyword within data, we find the data linked to said keyword, which is achieved by preparing an index where the trapdoor, associated to a keyword, links to a list of identifiers of the data that contains said keyword, all of which is encrypted. Curtmola et al. approach proposes that all nodes of all linked lists, associated to different keywords, should be part of a single array, in a scrambled order. The plaintext of each node includes the identifier of the data that contains the keyword, a pointer to the next node of the list as well as the key used to encrypt that node. The only thing the user needs then is the secret key of the first node of the list associated to the keyword they want to search for and the position of that node in the array. One of the downsides of the approach presented is that it is not dynamic, meaning the arrays would need to be updated whenever something is added or removed. Another issue, that affects performance, is that it is not parallelizable, meaning that the system can only focus on processing one thing at a time. This is so, since nodes are spread out randomly in the array and the only way to know where the next node of the list is by decrypting said node.

Kamara et al. [58] present a possible way to achieve **dynamic** searchable encryption by being able to track these operations in an efficient manner. Whenever data is added or deleted, the array positions that will suffer changes will be kept track of in a deletion array. Furthermore, a list of free nodes, keeping track of positions available on the search array is kept and used whenever new data is added to the server. Finally, the pointers of each node are updated through the usage of homomorphic encryption. Homomorphic encryption [59] is a type of encryption that allows a user to work and process data even while its encrypted, without the need to be decrypted. This technique allows for this approach to modify data, the pointers to next nods, while skipping decryption.

Another possibility for achieving dynamic searchable encryption is by building the inverted index while also working on it. This idea is presented by Hahn and Kerschbaum [60] and makes use of **both a forward index along with an inverted index**. The idea behind it is to first have a forward index, where the data/document is used as the index to find a keyword and an empty inverted index. When a keyword is first searched, the system first learns what the search token is, that will be used in the inverted index. Once that's done the keyword and its' data are added to the inverted index. If data/document that contains a certain keyword are added or deleted, the inverted index is updated accordingly. the forward index is used when a keyword is

first searched, which means a linear search time for the first search, however, once that's done, whenever users later search for the same keyword, the search token remains the same and users will access the inverted index instead.

Most searchable encryption schemes can only work with single keyword queries. Golle et al [61] first presented an approach that offered **multi-keyword searches**, achieved through the combination of different protocols that allow conjunctive keyword search. One of these approaches involves the intersection of multiple searches. If a user wants to search for documents that contain the keywords A and B, they first send a query for all documents that contain A and then do the same for B, and finally they check what documents are found on both results. This has the drawback of the server getting information from the searches which can later be used to infer about each document. Another approach is the usage of **meta-keywords**. Instead of sending a query that contains keyword A and then another for B, a query such as "A:B" is sent to retrieve documents with that meta-keyword. This requires that each document has associated to it all combinations of every keyword, represented in meta-keywords. This has the drawback of massive storage usage. Golle et al. present an approach that offers the benefits of the previous approaches while lowering the drawbacks which is achieved through **Boolean conjunctive query in linear performance**.

Later, Cao et al. [62] first proposed a **multi-keyword ranked searchable encryption** (MRSE), whose main idea was allowing users to search for multiple keywords and receive the most relevant results. This is achieved through "inner product similarity" which utilizes an algorithm, adapted from k-nearest neighbour technique, to enable the selection of the k-nearest database records between database record and query vector. This approach has the issue of using a static dictionary, requiring rebuilding it whenever a new keyword is added. Furthermore, it also does not account for the weight and access patterns of keywords when presenting the top results. As such, R. Li et al [63] proposed a new scheme, MKQE, to tackle and overcome MRSE's faults.

More recently, Liu et al [64] present a searchable encryption scheme named Searchable Encryption based on Efficient Privacy-preserving Outsourced calculation framework with Multiple keys (SE-EPOM), capable of **multi-keyword search in a multi-server architecture**, while also allowing for **multiple writers**. This scheme is based on a subset decision mechanism, also developed by the authors, with the goal of determining whether one input is the subset of another input set.

Regarding forward and backward privacy, topics that have also seen research efforts in recent years in the context of searchable encryption, forward privacy means that when adding a document, no information is revealed about its' keywords, whether what keywords it contains or if they have been searched. As for backward privacy, it must guarantee that searches do not leak information about keywords that have already been deleted. Sefanov et al. (2014) introduced these concepts in searchable encryption and proposed a scheme that used **forward privacy**. The concept was

improved by Bost et al. [65] who focused on researching **backward privacy**, which was until then overlooked and proposed different schemes with both forward and backward privacy. More recently, Bakas and Michalas [66] applied **forward and backward privacy** in a **multi-client cloud** environment.

Searchable Encryption usage in Cyber Threat Intelligence sharing is a topic which is still very overlooked; hence, there is limited research on the topic. INESC TEC has achieved two different publications on this topic, both authored by Fernandes et al. In the first publication [67] it is mentioned that although the Malware Information Sharing Platform (MISP) allows for sharing of CTI, it presents limitations in the way the CTI sharing can be controlled and searched within groups of entities while maintaining its confidentiality. As such, a prototype that allowed for CTI to be shared between entities through a proxy API, which also connects to a MISP instance, is proposed. As for the second publication [68], the performance of the previous prototype is evaluated and improved, with the results being presented.

7.1.2 Decentralization

Current CTI-sharing solutions present drawbacks, one of them being: **Single Point of Failure**; meaning that the system is dependent on one server to work, which hosts the reverse index containing data that will be exchanged. If this server goes down, the whole system suffers a failure. Another drawback is their inability to support the **dynamic** creation and operation of autonomous CTI-sharing groups.

A way to overcome the Single Point of Failure problem is by no longer being dependent on a single server. This entails the existence of a distributed index, which also requires multiple systems hosting indexes. Such a concept is explored by Cai et al. [69], where an encrypted decentralized storage architecture, which also allows for **private keyword search**, is proposed. Blockchain is used as the decentralized data storage platform of choice. In this solution there will exist two types of peers: i) client peers, responsible for outsourcing files and indexes, as well as verifying keyword searches and ii) storage peers, responsible for returning search results. To minimize search latency, encrypted files and their encrypted index are stored in the same peer.

More recently, Sultan et al. [70] presents a scheme, to be used in the context of Internet of Things, that allows for **multi-client usage**, offers **forward and backward privacy** and a **distributed index** through distributed hash tables.

As referenced beforehand, one way of achieving this is through Blockchain. The concept was first explored by Haber and Stornetta [71], with the objective of timestamping digital documents through cryptographically secure blocks of chains. This concept would go under the radar until some years later, when Nakamoto [72] and the introduction of cryptocurrencies, through Bitcoins, would cause the concept

to gain much traction. Furthermore, this was also the paper that popularized the term "Blockchain". In this paper, Nakamoto presents a decentralized and distributed electronic payment system, without the intervention of third parties. Since then, Blockchain has been a topic of study in several industry sectors, such as:

- Supply Chains: Palamara [73] analyses what advantages Blockchain can bring to supply chain and proposes a solution for tracking products to a company in the chocolate industry. Canavari et al. [74] also references the tracking of meats in the UK, allowing for transparency, and how it has developed. It also studies the underdevelopment of Blockchain, in comparison, for fresh produce, while mentioning what is making Blockchain fall behind.
- IoT: Wang et al. (2020) presents the benefits of Blockchain to IoT an Industrial IoT, while also introducing the main features of Blockchain applied to IoT for Industry 4.0. Mathur et al. [75] reviews the advantages Blockchain brings to IoT and introduces the main aspects of using Blockchain in IoT as well as the main implementation challenges.
- Healthcare: As with all these other pieces of research, in healthcare, Blockchain can be used in order to safely store information of patients, Ejaz et al. [76], Zhang et al. [77], validate and check integrity of all records, Tanwar et al. [78], Pham et al. [79] as well as ensuring the transparency of system communications, Hathaliya et al. [80].

Another way to achieve decentralization is through the usage of Torrent, a P2P protocol, also known as BitTorrent. It is a protocol mostly used for file transfers and it entails the usage of "torrent" files that contain metadata of the data that will be shared as well as what other systems have that information (peers).

Released in 2008, the BitTorrent[10] protocol has, mostly, seen use in P2P overlay networks (networks built on top of existing networks). P2P overlay networks have seen some research – Polar et al. [81] proposed a newer design at the time for P2P overlay networks in fiber optic communications; Caubet et al. [82] presented a new protocol that allowed for better security in these networks, while maintaining the anonymity; Srivastava and Ahmad [83] also proposed a new probabilistic gossip-based secure protocol to track faulty peers. As can be seen in the previous papers, the security of the BitTorrent protocol was often called into question. As such, in 2020 BitTorrent v.2[11] was released with a focus on improving on the security faults of the previous version.

The BitTorrent protocol itself was researched for application in multiple scenarios. Lee and Nakao [84] presented approaches and results on how to achieve a more efficient usage of BitTorrent to lower traffic, while satisfying both the users and the ISPs.

[10] https://www.bittorrent.org/beps/bep_0003.html
[11] https://blog.libtorrent.org/2020/09/bittorrent-v2/

Kopiczko et al. [85] presented "StegTorrent" a steganographic method based on BitTorrent, allowing for users to share secret information within data being shared in such a way that it is indiscernible and difficult for others to detect. Although the protocol is not used, Neuner et al. [86] proposed the usage of data from BitTorrent networks to create hash databases for digital forensics. To detect if a system has a certain file, it first needs to have a parameter in its' database that tells it what to search for. But if the files in the system being searched have suffered some kind of change, the parameter used for searching may not be usable. Since BitTorrent is a protocol used mostly for file transfers, to ensure the integrity of the files, it works with large quantities of hash values, hence the choice of BitTorrent to feed the hash database.

While Blockchain is currently a topic under more research and development, the adoption of an approach similar to the one of the BitTorrent protocol would solve the single point of failure, as each sharing group could have its own index, but also enables a more dynamic environment with support for the establishment of ad hoc sharing groups.

7.1.3 PRIVATEER's Innovation in CTI

The PRIVATEER proposed solution innovates upon the research in Searchable Encryption by presenting a way to exchange data with two particularities: Firstly, it is a decentralized system, with all entities having shared encrypted indexes, which synchronize between each other, and overcomes the single point of failure. And secondly, it allows for user-generated sharing groups to control information – in each group an entity has a different index linked to it, entities can only share information with other entities that belong to the same shared group and an entity can set up different policies for data exchange for different shared groups, further controlling the flow of information.

7.2 Protocol description

In order to ensure the security of the solution, the indexing and the search querying must be confidential. There does not exist in scientific literature a standard security model for searchable encryption, thus, for the threat model, an honest-but-curious server is adopted. This entails that a server follows the rules, however, it is interested in learning sensitive information through analysis of search queries. Furthermore, we also assume the existence of a third-party malicious actor that will attempt to read, alter or delete data.

The solution must also ensure that it can resist common attack vectors in the searchable encryption field, during the search phase, such as Chosen-Keyword Attack (CKA) and Known Keyword Attack (KGA). A CKA attack happens, in a searchable encryption scenario, when a malicious actor gains information of the stored data by

retrieving the plain text keyword from an encrypted keyword. A way to overcome this issue is by ensuring that the server does not know any keyword, thus, it becomes unfeasible for a malicious actor to figure out whether a certain ciphertext contains a certain keyword. A KGA attack happens, in a searchable encryption scenario, if the malicious actor is the one that generated all of the cyphertexts of all keywords. With the knowledge of one trapdoor, the attacker can search the cyphertexts for matching results and when a cyphertext that contains the keyword is found, the attacker can guess what keyword is related to the trapdoor.

Index confidentiality is achieved through HMAC running at client-side, using secret keys that only the entities know, making it unlikely that any other group of entities is capable of generating the trapdoors. Since HMAC is a one-way mathematical function, it is also not feasible for a malicious actor to obtain the original keywords. The solution is resistant to CKA attacks, thanks to HMAC as well. Since an attacker requires the original keywords in order to be able to retrieve any information and since HMAC is a one-way function generated with a secret key, that only the participating entities will hold, it becomes unfeasible for the attacker to follow this approach. The solution is also resistant to KGA attacks since only participating entities will be able to generate keyword trapdoors.

In the sequence diagrams of this section, three participants are presented: two entities and a shared index. The entity represents an organization or individual that wants to share or receive CTI data through our CTI sharing proxy API (end-user) and the shared index represents a database that was properly set up to work with our proxy API – inverse index, encrypted information that can be queried through trapdoors. In terms of PRIVATEER the consumer of the CTI information can be the Privacy-aware Orchestrator and the Level of Trust Assessment component.

7.2.1 Set Up

Presented in Figure 10 is the setup process: When an entity wants to begin communicating with another entity, it starts by exchanging UUIDs and public keys **(step 1)**. Then each entity adds the other entity's information to their local database **(step 2 and 4)** and associates the public key received to the new record **(step 3 and 5)**. Once that's done a peer validation process is performed **(step 6)** and is presented in Figure 11.

In this process, presented in Figure 11, a nonce is generated first **(step 1)**. The plaintext message, which is a combination of the UUID of the sender, a timestamp and the nonce generated is prepared **(step 2)**. This message is encrypted with the public key of the receiver **(step 3)**. Another message is created which consists of a combination of the UUID of the sender, timestamp and the encrypted message, done to ensure integrity **(step 4)**. This message is sent to the other entity **(step 5)**. The receiver starts by splitting the message into a UUID, timestamp and an encrypted message **(step 6)**. The encrypted message is decrypted with the entity's private key **(step 7)**. Once that's

done, the contents of the decrypted message are split, once again into a UUID, timestamp and a nonce **(step 8)**. The two UUIDs are compared to verify if they match **(step 9)**. If they don't then the integrity of the message is called into question. Afterwards, a new message is prepared containing the receiver's UUID, a timestamp and a nonce **(step 10)**. This message is encrypted with the sender's public key **(step 11)**. Another message is prepared with the UUID of the receiver, a timestamp and the encrypted message **(step 12)**. This message is sent back to the sender **(step 13)**.

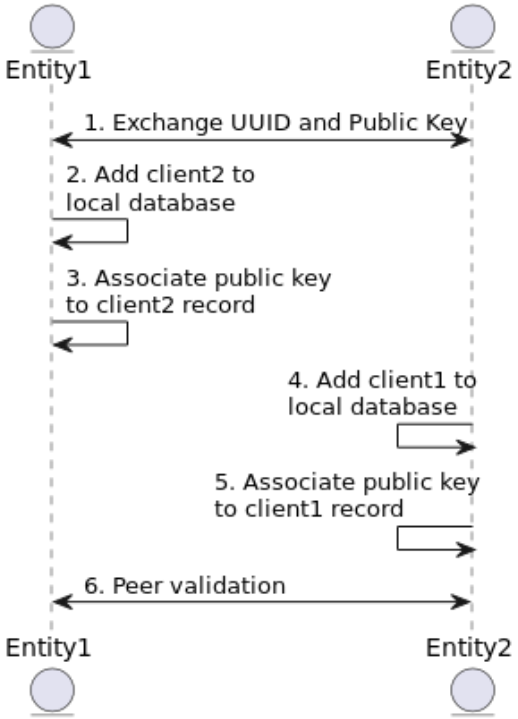

Figure 10 - Set Up

The process is very similar to what the receiver did to the sender's message. It's split into different parts and decrypted **(steps 14, 15, 16)**, and the UUID and nonces are compared to verify that they match to ensure integrity **(steps 17, 18)**. If no errors occur, the peer status is changed to OK **(step 19)**.

7.2.1 MISP Data Sync

When preparing the data synchronisation between MISP instances (see Figure 12), the entities must first prepare a daily job to sync data between each other **(step 1, 2)** and setup policies, to define what information can and can't be shared **(step 3, 4)**. When the daily job starts, the API proxy sends a request for a MISP data sync to the other entity **(step 5, 7)**, and, if no errors occur, a MISP data sync response with the data requested is sent back **(step 6, 8)**.

All data sync processes are run through MISP's API.

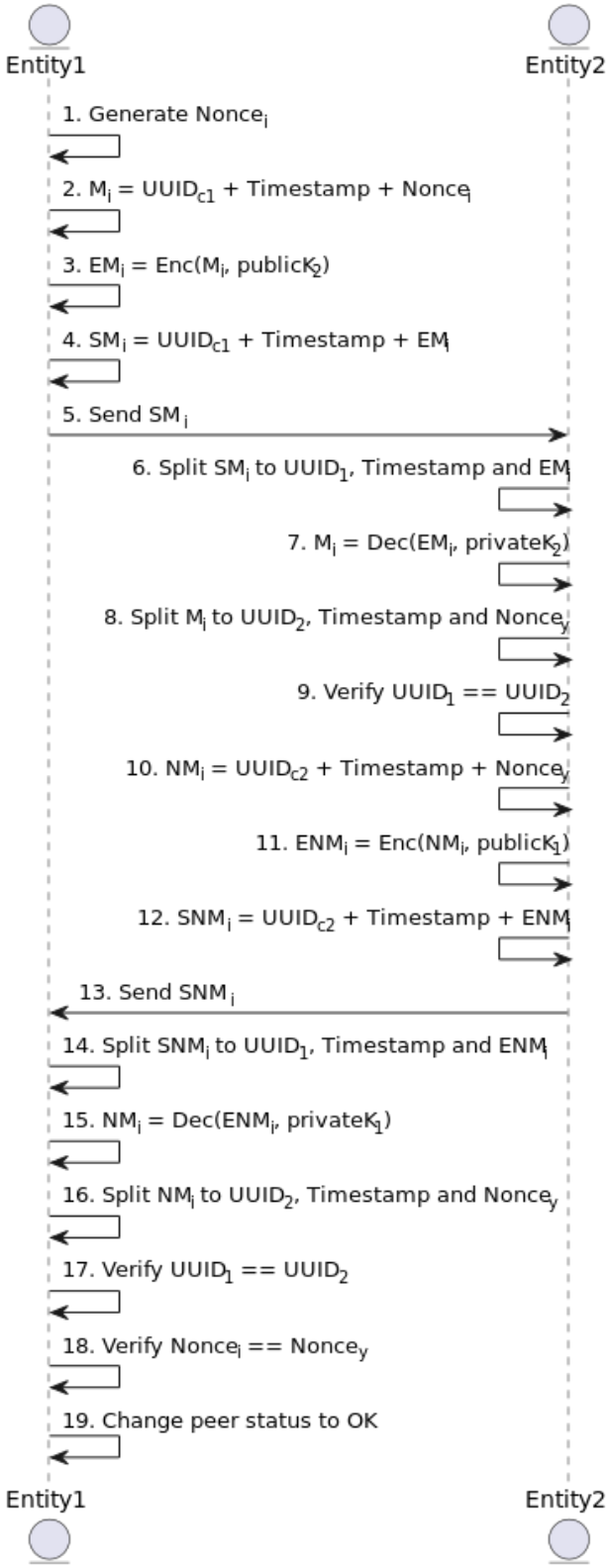

Figure 11 - Peer Validation

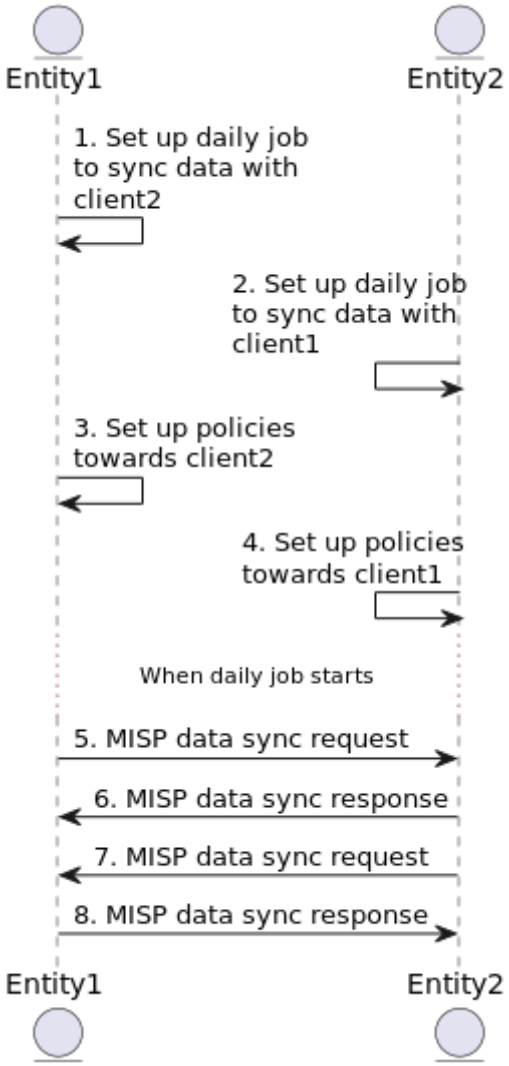

Figure 12 - MISP Data Sync

7.2.2 Shared Index

To start, a shared group needs to be created on each entity **(steps 1 and 2)**, as illustrated in Figure 13. Then the entities that are part of the shared group need to be added to the shared group, which is done by associating the UUID to it **(steps 3 and 4)**. Afterwards, the secret key of the shared group is generated **(step 5)** and presented in Figure 14.

As illustrated in Figure 14, the generation of the secret key is done by first retrieving the group hash, or if it doesn't exist, a new one is generated **(step 1)**. A message is prepared containing the UUID of the sender, a timestamp, the group name, a list of the entities part of the group and the group hash **(step 2)**. This message is encrypted with the public key of the receiver **(step 3)**. Another message is prepared containing the UUID of the sender, a timestamp and the encrypted message **(step 4)**. This message is sent to the receiver **(step 5)**.

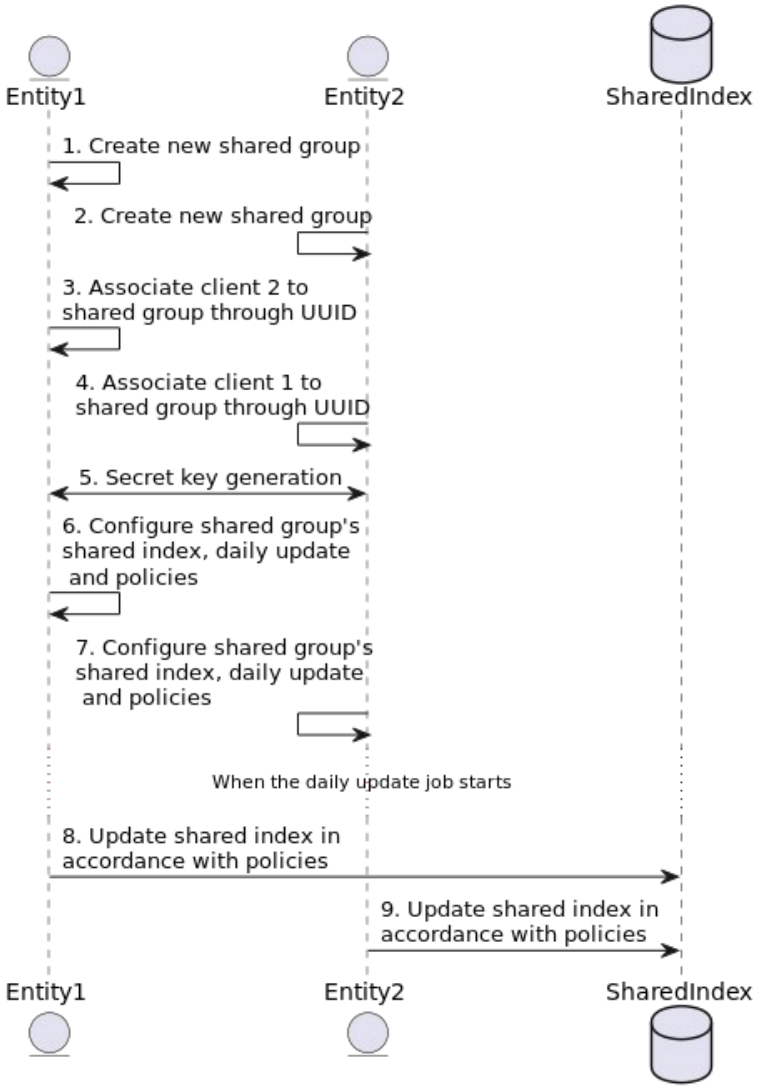

Figure 13 - Shared Group Set Up

The receiver splits the message into three parts: the UUID of the sender, a timestamp and an encrypted message **(step 6)**, which is decrypted with the private key of the receiver **(step 7)**. This decrypted message is split into five parts: another UUID of the sender, a timestamp, the group's name, the list of participants of the group and the group's hash **(step 8)**. The UUIDs retrieved are compared to verify if they match **(step 9)**. The receiver then checks if they belong to the group that is referenced in the message received **(step 10)**. If so, the information received of the group is compared with the information available on the receiver's end, once again to verify if it matches **(step 11)**. If no errors occur, the receiver retrieves **(step 12)** and adds their hash to the group hash **(step 13)**. A new message containing the UUID of the receiver, a timestamp and the receiver's hash is prepared **(step 14)**. This message is encrypted with the

sender's public key **(step 15)**. A new message is prepared containing the UUID of the receiver, a timestamp and the encrypted message **(step 16)**. This last message is sent back to the sender **(step 17)**.

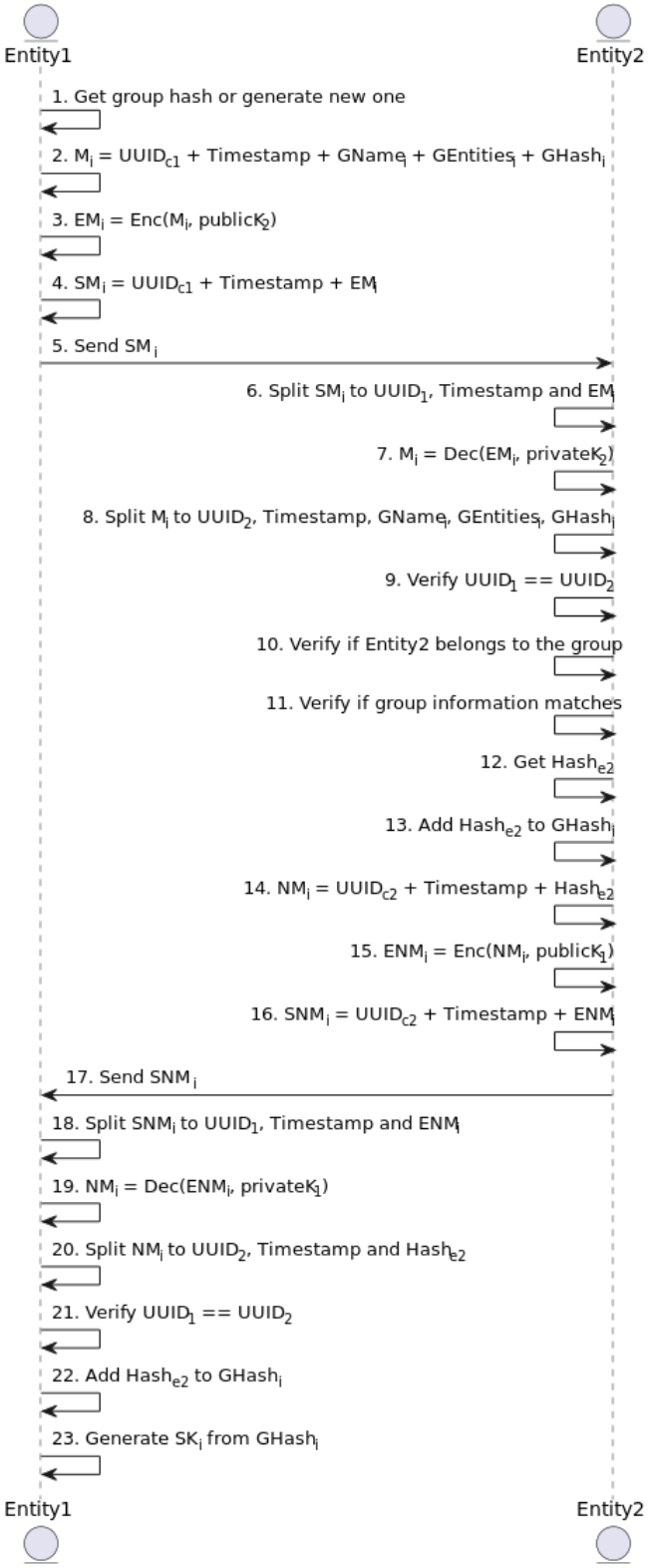

Figure 14 - Secret Key Generation

The sender splits the message into the receiver's UUID, timestamp and an encrypted message **(step 18)**. The encrypted message is decrypted with the sender's private key **(step 19)**. The decrypted text is split into another UUID, timestamp and a hash **(step 20)**. The UUIDs are compared and checked to see if they match **(step 21)**. If all goes well, the hash of the receiver is added to the group hash **(step 22)**. The secret key is generated from the group hash **(step 23)**.

Returning to Figure 13, with the new secret key generated, the shared group's configuration is prepared. The entity needs to set up the shared index's location, the daily update job and set up the policies, or in other words, what information can be shared using the index **(steps 6 and 7)**. When the daily update job begins, the API updates the shared index **(steps 8 and 9)**.

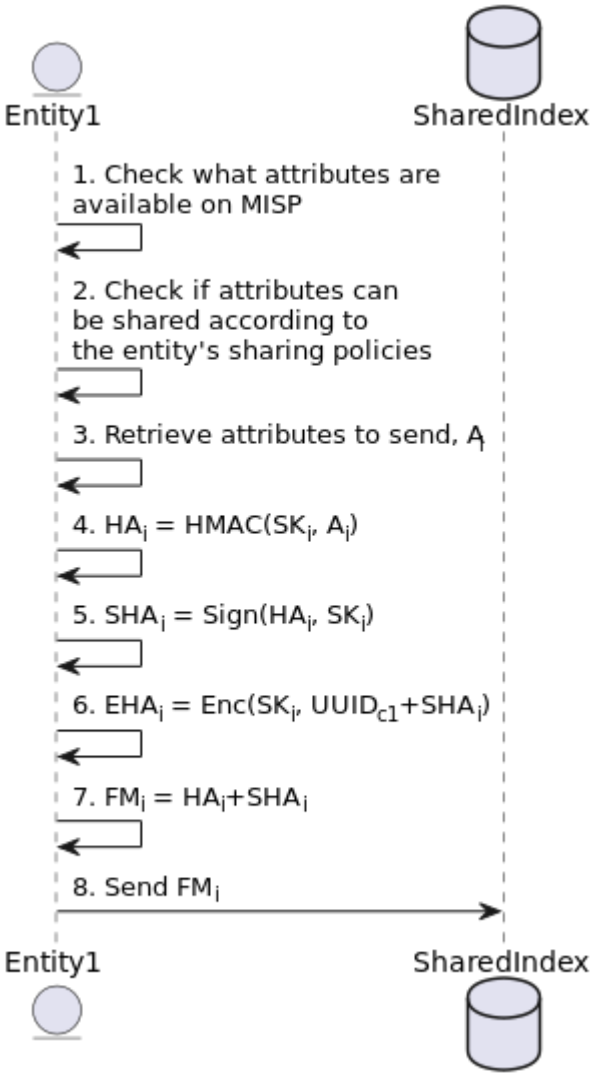

Figure 15 - Shared Index Update

With this update the API starts by checking what attributes are available on the MISP **(step 1)** as illustrated in Figure 15 and whether these attributes can be shared

according to the entity's sharing policies **(step 2)**. If so, the information about those attributes is retrieved **(step 3)**. A trapdoor, or hash, is generated from the shared group's secret key and the attribute data **(step 4)**. A new hash is generated, differing from the previous one because it's signed with the shared group's secret key **(step 5)**. This new hash is appended to the UUID of the sender and encrypted with the shared group's secret key **(step 6)**. A message is created from the trapdoor **(step 7)** and the encrypted hash and sent to the shared index **(step 8)**.

7.3 Plan for development

As mentioned, one of the main concerns of many searchable encryption schemes is the reliance on a single server, hence they suffer from single point of failure. If the server hosting the encrypted shared index fails in some way, the whole system is disrupted. The current CTI Sharing proxy API solution developed in PRIVATEER is also susceptible to single point of failure and as such, future plans include overcoming this risk by implementing encrypted shares index decentralization. This requires that there exist multiple encrypted shared indexes, that are synchronized with each other, so if one fails, users can still use one of the other ones available.

Regarding decentralization, different approaches were considered: Blockchain and Distributed Hash Tables were some of the possible choices. However, after some research, it became clear that this goal could be achieved with an implementation inspired by the BitTorrent protocol that, overall, adapted more smoothly to the current architecture, without the need to implement too many new modules or components. This approach fits well with the current solution, since the current solution has the functionality to create and join shared groups and when doing so it requires the input of information about the other entities that belong to the same group, to ensure that the shared key generated when exchanging data is correct, with one of those pieces of information being the IP address of the entities – hence, the only requirement to implement decentralization becomes that all entities now have a database running on their system, or access to one, which hosts the multiple encrypted indexes for their groups and that all entities synchronize these encrypted indexes whenever another entity updates theirs.

Figure 16 presents the idea – each entity needs to be part of at least one shared group to be able to share information with others, furthermore, each entity is required to have a MISP instance connected to the proxy API to store the CTI data. The proxy API feeds the encrypted index in the system according to two conditions: the MISP instance receives new CTI data (from external sources) and the policies that were applied for the shared group fit this new data. This causes the encrypted index to be updated and then a synchronization notification is sent to all other peers, to update their encrypted index with the new information.

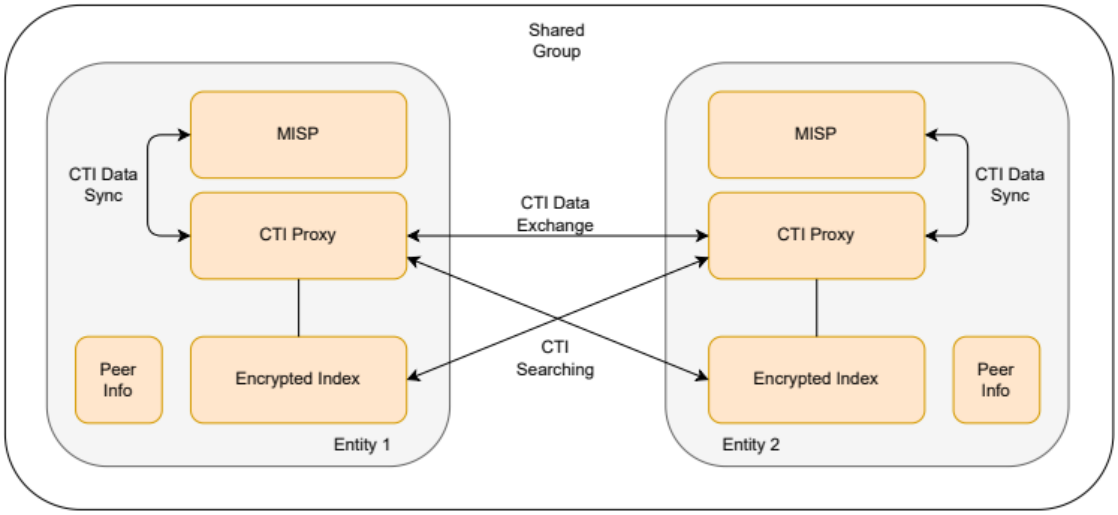

Figure 16 - CTI Internal Architecture

Another possibility is the development of a micro-service that supports this new distributed index for searches of CTI but also enables them to be performed in lower-performance devices. This entails the usage of the API without a dependency on a MISP instance – thus these users themselves do not host any CTI data but can communicate with other entities to perform searches.

8 Conclusions

In the context of future B5G, achieving Zero Trust based security requires a multifaceted approach. Ensuring the integrity and resilience of both the services and the underlying infrastructure is paramount for protecting against sophisticated cyber threats that exploit vulnerabilities in interconnected services or networks. Evidently a comprehensive approach to security, encompassing both remote and local attestation, identity verification, and authentication, is essential. Towards this direction PRIVATEER has addressed these challenges by integrating hardware-enabled Trusted Execution Environments (TEEs), privacy-preserving mechanisms, and Distributed Ledger Technology (DLT), for both data exchange and identity management through Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs). The present deliverable delved into the details and internal architecture and functionalities of WP5. This WP provides the enablers in terms of attestation-related evidence, leveraged by the Level of Trust Assessment framework. The PRIVATEER architecture represents a comprehensive framework for enhancing security, privacy, and trust in distributed and complex B5G environments.

The incorporation of hardware-enabled Trusted Execution Environments (TEEs), such as Intel SGX, has established a strong Root of Trust (RoT) within the system, providing the foundation for isolated execution of critical workloads. Secure deployment mechanisms, such as Gramine and enclace-cc for confidential container launching and eBPF tracers for extraction of runtime evidence, enhance security by enabling configuration integrity verification. µProbes deployed within the containers further bolster security by providing insights into the configuration integrity of containerized applications. Additionally, edge accelerators (i.e., FPGAs) are also monitored through dedicated attestation agents, ensuring that the bitstream has not been altered.

Integration with Distributed Ledger Technology (DLT) has enhanced transparency, immutability, and accountability of trust-related data exchange. The Secure Oracle's role in smart contract execution facilitates trust assessment processes by validating attestation reports and generating comprehensive smart contracts. These contracts encapsulate essential information required for trust assessment, ensuring transparency and accountability throughout the system. It shall be noted that apart from secure and auditable data exchange, DLT is leveraged for identity management in PRIVATEER, supporting the notion of Self Sovereign Identity (SSI). Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs) enable secure and decentralized identity verification, contributing to a trustworthy ecosystem.

In addition to the above, PRIVATEER has employed privacy-preserving mechanisms, for protecting sensitive information from unauthorized disclosure while allowing access to essential trust-related data to external parties. Towards this direction, the

Trust Exposure Layer along with Attribute-based Access Control (ABAC) mechanisms is supported, ensuring that only authorized parties have access to information available on the ledger. By implementing Attribute-based Encryption (ABE), PRIVATEER ensures the confidentiality and privacy in data access and exchange processes in the off-chain storage too. Furthermore, the implementation of a privacy-preserving CTI sharing mechanism has been taken into account to improve the security level of an entity, specifically a Mobile Network Operator (MNO). This mechanism enables the exchange of threat information, thereby facilitating compliance with the existing threat landscape.

The aforementioned components have offered a strong foundation for secure and trustworthy interactions across distributed environments. The upcoming Release B of the PRIVATEER platform will include enhancements and additional features, resulting in the completion of the framework's final view.

9 References

[1] Hexa-X A flagship for B5G/6G vision and intelligent fabric of technology enablers connecting human, “Deliverable D1.4 Hexa-X architecture for B5G/6G networks – final release,” 2023.

[2] PRIVATEER, “Deliverable 2.2: Use cases, requirements and design report,” 2023.

[3] PRIVATEER, “Deliverable 4.1: Privacy-aware slicing and orchestration enablers - Rel. A,” 2024.

[4] “ETSI TS 133 501 V15.4.0, 5G; Security architecture and procedures for 5G System,” 2019.

[5] “RFC7285: "Application-Layer Traffic Optimization (ALTO) Protocol",” Internet Engineering Task Force (IETF), 2014.

[6] “Internet-Draft: Considering ALTO as IETF Network Exposure Function,” Internet Engineering Task Force (IETF), 2023.

[7] Intel, “Whitepaper: Overview on Signing and Whitelisting for Intel Software Guard Extension (Intel SGX) enclaves,” 2018.

[8] “RFC 7223: A YANG Data Model for Interface Management,” Internet Engineering Task Force (IETF), 2020.

[9] PRIVATEER, “Deliverable 2.4: PRIVATEER framework demonstrator – Rel. B,” 2025.

[10] Hyperledger Foundation, "Hyperledger Indy," [Online]. Available: https://www.hyperledger.org/projects/hyperledger-indy. [Accessed 14 March 2024].

[11] R. Román, R. Arjona and I. Baturone, “A lightweight remote attestation using PUFs and hash-based signatures for low-end IoT devices,” *Future Generation Computer Systems,* vol. 148, pp. 425-435, 2023.

[12] U. Javaid, M. N. Aman and B. Sikdar, “Defining trust in IoT environments via distributed remote attestation using blockchain,” in *International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (Mobihoc '20)*, 2020.

[13] PCI-SIG, “TEE Device Interface Security Protocol (TDISP),” 2022.

[14] “ ISO/IEC TS 5723:2022, Trustworthiness Vocabulary,” 2022.

[15] A. Martin, “The ten-page introduction to Trusted Computing.,” 2008.

[16] E. C.-L. Raghuram Yeluri, “Building the Infrastructure for Cloud Security: A Solutions View,” *Springer Nature,* p. 244, 2014.

[17] ETSI, “Network Functions Virtualisation (NFV): Trust: Report on Attestation Technologies and Practices for Secure Deployments, ETSI GR NFV-SEC 007 V1.1.1,,” 2017.

[18] L. Ferro, Container Attestation with Linux IMA namespace. PhD diss., Torino, Italy: Politecnico di Torino, 2023.

[19] M. De Benedictis and A. Lioy, “Integrity verification of Docker containers for a lightweight cloud environment.,” *Future generation computer systems,* vol. 97, pp. 236-246, 2019.

[20] A. Proulx, J.-Y. Chouinard, P. Fortier and A. Miled, “A survey on fpga cybersecurity design strategies,” *ACM Transactions on Reconfigurable Technology and Systems,* vol. 16, no. 2, p. 1–33, 2023.

[21] N. N. Anandakumar, M. S. Hashmi and M. Tehranipoor, “FPGA-based Physical Unclonable Functions: A comprehensive overview of theory and architectures,” *Integration,* vol. 81, p. 175–194, 2021.

[22] F. Verbauwhede and I. Turan, “Trust in FPGA-accelerated cloud computing,” *ACM Computing Surveys (CSUR),,* vol. 53, no. 6, p. 1–28, 2020.

[23] N. Khan, A. Silitonga, B. Pachideh, S. Nitzsche and J. Becker, “Secure local configuration of intellectual property without a trusted third party,” in *Applied Reconfigurable Computing (ARC) 15th International Symposium*, Darmstadt, 2019.

[24] B. Hong, H.-Y. Kim, M. Kim, T. Suh, L. Xu and W. Shi, “Fasten: An fpga-based secure system for big data processing,” *IEEE Design & Test,,* vol. 35, no. 1, p. 30–38, 2017.

[25] S. Kuhn and M. G. Drimer, “A protocol for secure remote updates of FPGA configurations,” in *International Workshop on Applied Reconfigurable Computing*, 2009.

[26] M. Zhao, M. Gao and C. Kozyrakis, “Shef: Shielded enclaves for cloud fpgas,” in *27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2022.

[27] J. Vliegen, M. M. Rabbani, M. Conti and N. Mentens, “SACHa: Self-attestation of configurable hardware,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2019.

[28] Y. Wang, X. Chang, H. Zhu, J. Wang, Gong, Y and L. Li, “Towards Secure Runtime Customizable Trusted Execution Environment on FPGA-SoC,” *IEEE Transactions on Computers,* 2024.

[29] M. M. Ahmadi, F. Khalid, R. Vaidya, F. Kriebel, A. Steininger and M. Shafique, “‘SHIELD: An Adaptive and Lightweight Defense against the Remote Power Side-Channel Attacks on Multi-tenant FPGAs,” arXiv preprint arXiv:2303. 06486, 2023.

[30] Y. Luo, S. Duan and X. Xu, “FPGAPRO: A defense framework against crosstalk-induced secret leakage in FPGA,” *ACM Transactions on Design Automation of Electronic Systems (TODAES),* vol. 27, no. 3, p. 1–31, 2021.

[31] T. M. La, K. Matas, N. Grunchevski, K. D. Pham and D. Koch, “FPGADefender: Malicious self-oscillator scanning for Xilinx Ultrascale+ FPGAs,” *ACM Transactions on Reconfigurable Technology and Systems (TRETS),* vol. 13, no. 3, p. 1–31, 2020.

[32] PRIVATEER, “Deliverable 2.1: 6G threat landscape and gap analysis ,” 2023.

[33] W. Zou and e. al, “Smart contract development: Challenges and opportunities,” *IEEE transactions on software engineering,* vol. 47, no. 10, pp. 2084-2106, 2019.

[34] Z. Hussein, M. A. Salama and S. A. El-Rahman, “Evolution of blockchain consensus algorithms: a review on the latest milestones of blockchain consensus algorithms,” *Cybersecurity ,* vol. 6, no. 30, 2023.

[35] N. R. Kasi, R. S and M. Karuppiah, “Chapter 1 - Blockchain architecture, taxonomy, challenges, and applications,” in *Hybrid Computational Intelligence for Pattern Analysis in Blockchain Technology for Emerging Applications*, 2022.

[36] “Hyperledger Fabric Whitepaper,” Hyperledger Foundation.

[37] “Whitepaper: An Introduction to Hyperledger,” Hyperledger, 2018.

[38] F. Zhang, E. Cecchetti, K. Croman, A. Juels and E. Shi, “Town Crier: An Authenticated Data Feed for Smart Contracts,” in *ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*, 2016.

[39] C. Mazzocca, A. Acar, S. Uluagac, R. Montanari, . P. Bellavista and M. Conti, "A Survey of Decentralized Identifiers and Verifiable Credentials," 2024.

[40] K.-L. Tan, C.-H. Chi and K.-Y. Lam, "Analysis of Digital Sovereignty and Identity: From Digitization to Digitalization," 2022.

[41] R. Soltani, U. T. Nguyen and A. An, "A Survey of Self-Sovereign Identity Ecosystem," *Security and Communication Networks,* p. 26, 2021.

[42] M. Takaoğlu, T. Dursun and A. Doğan, "The Impact of Self-Sovereign Identities on CyberSecurity," 2023.

[43] A. Satybaldy, A. Subedi and M. Nowostawski, "A Framework for Online Document Verification Using Self-Sovereign Identity Technology," *Sensors,* p. 22, 2022.

[44] C. Sehlke, "Transforming a Digital University Degree Issuance Process Towards Self-Sovereign Identity," 2022.

[45] Y. Bai, H. Lei, S. Li, H. Gao, J. Li and L. Li, "Decentralized and Self-Sovereign Identity in the Era of Blockchain: A Survey," in *2022 IEEE International Conference on Blockchain (Blockchain)*, 2022, pp. 500-507.

[46] P. Bolte, "Self-sovereign Identity: Development of an Implementation-based Evaluation Framework for Verifiable Credential SDKs," 2021.

[47] ENISA, "Digital Identity: Leveraging the SSI Concept to Build Trust," 2022.

[48] World Wide Web Consortium, "Verifiable Credentials Data Model v2.0," 2024. [Online]. Available: https://www.w3.org/TR/vc-data-model-2.0/. [Accessed 14 March 2024].

[49] F. Schardong and R. Custódio, Self-Sovereign Identity: A Systematic Map and Review, 2021.

[50] L. Stockburger, G. Kokosioulis, A. M. Mukkamala, R. R. Mukkamala and M. Avital, "Blockchain-enabled Decentralized Identity Management: The Case of Self-sovereign Identity in Public Transportation.," *Blockchain: Research and Applications,* vol. 2, p. 18, 2021.

[51] World Wide Web Consortium, "Decentralized Identifiers (DIDs) v1.0," 2022. [Online]. Available: https://www.w3.org/TR/did-core/. [Accessed 14 March 2024].

[52] Hyperledger Foundation, "Aries Hyperledger," [Online]. Available: https://www.hyperledger.org/projects/aries. [Accessed 04 April 2024].

[53] Hyperledger Aries Cloud Agent - Python, "GitHub," [Online]. Available: https://github.com/hyperledger/aries-cloudagent-python. [Accessed 04 April 2024].

[54] M. Sabadello and A. Horvat, "Identifiers and Discovery Working Group," Decentralized Identity Foundation, [Online]. Available: https://identity.foundation/working-groups/identifiers-discovery.html. [Accessed 19 March 2024].

[55] D. X. Song, D. Wagner and A. Perrig, "Practical techniques for searches on encrypted data," in *IEEE symposium on security and privacy (IEEE S&P)*, 2000.

[56] E.-J. Goh, "Secure indexes," Cryptology ePrint Archive, 2003.

[57] R. Curtmola, J. Garay, S. Kamara and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in *13th ACM conference on Computer and communications security*, 2006.

[58] S. Kamara, C. Papamanthou and T. Roeder, "Dynamic searchable symmetric encryption," in *ACM conference on Computer and communications security*, 2012.

[59] X. Yi, R. Paulet and E. Bertino, "Fully Homomorphic Encryption," *Homomorphic Encryption and Applications,* pp. 47-66, 2014.

[60] F. Hahn and F. Kerschbaum, "Searchable encryption with secure and efficient updates," in *ACM SIGSAC Conference on Computer and Communications Security*, 2014.

[61] P. Golle, J. Staddon and B. Waters, "Secure conjunctive keyword search over encrypted data," in *Applied Cryptography and Network Security: Second International Conference (ACNS 2004)*, Yellow Mountain, China, 2004.

[62] N. Cao, C. Wang, M. Li, K. Ren and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Transactions on parallel and distributed systems,* vol. 25, no. 1, pp. 222-233, 2013.

[63] R. Li, Z. Xu, W. Kang, K. C. Yow and C.-Z. Xu, "Efficient multi-keyword ranked query over encrypted data in cloud computing," *Future Generation Computer Systems,* vol. 30 , pp. 179-190, 2014.

[64] X. Liu, G. Yang, W. Susilo, J. Tonien, X. Liu and J. Shen, "Privacy-preserving multi-keyword searchable encryption for distributed systems.," *IEEE Transactions on Parallel and Distributed Systems,* vol. 32, no. 3, pp. 561-574., 2020.

[65] R. Bost, M. Brice and O. Ohrimenko, "Forward and backward private searchable encryption from constrained cryptographic primitives," in *ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[66] A. Bakas and A. Michalas, "Nowhere to leak: Forward and backward private symmetric searchable encryption in the multi-client setting (Extended Version)," Cryptology ePrint Archive, 2021.

[67] R. Fernandes, P. Pinto and A. Pinto, "Controlled and Secure Sharing of Classified Threat Intelligence between Multiple Entities," in *IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, 2021 .

[68] R. Fernandes, S. Bugla, P. Pinto and A. Pinto, "On the performance of secure sharing of classified threat intelligence between multiple entities," *Sensors ,* vol. 23, no. 2, p. 914, 2023.

[69] C. Cai, X. Yuan and C. Wang, "Towards trustworthy and private keyword search in encrypted decentralized storage," in *IEEE International Conference on Communications (ICC)*, 2017.

[70] N. H. Sultan, K.-K. S. Y. Tran, S. Lai, V. Varadharajan, S. Nepal and X. Yi, "A Multi-Client Searchable Encryption Scheme for IoT Environment," arXiv preprint arXiv:2305.09221 , 2023.

[71] S. Haber and W. S. Stornetta, " How to time-stamp a digital document," *Springer Berlin Heidelberg, ,* 1991.

[72] S. NAKAMOTO, " Bitcoin: A peer-to-peer electronic cash system," 2008.

[73] P. Palamara, "Tracing and tracking with the blockchain," 2016.

[74] R. K. Osei, M. Canavari and M. Hingley, "An exploration into the opportunities for blockchain in the fresh produce supply chain," 2018.

[75] M. Shikha, K. Anshuman, G. Gürkan, B. M. Kumar and L. Madhusanka, "A Survey on Role of Blockchain for IoT: Applications and Technical Aspects," *Computer Networks,* vol. 227 , p. 109726, 2023.

[76] M. Ejaz, T. Kumar, I. Kovacevic, M. Ylianttila and E. Harjula, "Health-blockedge: Blockchain-edge framework for reliable low-latency digital healthcare applications," *Sensors ,* vol. 21, no. 7, p. 2502, 2021.

[77] P. Zhang and B. Maged N. Kamel, "Blockchain solutions for healthcare," *Precision medicine for investigators, practitioners and providers,* pp. 519-524, 2020.

[78] S. Tanwar, K. Parekh and R. Evans, "Blockchain-based electronic healthcare record system for healthcare 4.0 applications," *Journal of Information Security and Applications,* vol. 50 , p. 102407, 2020.

[79] H. L. Pham, T. H. Tran and Y. Nakashima, "A secure remote healthcare system for hospital using blockchain smart contract," in *IEEE globecom workshops (GC Wkshps),*, 2018.

[80] J. Hathaliya, P. Sharma, S. Tanwar and R. Gupta, "Blockchain-based remote patient monitoring in healthcare 4.0," in *IEEE 9th international conference on advanced computing (IACC)*, 2019.

[81] A. Polar, M. Bunruangses, K. Luangxaysanam, S. Mitatha and P. P. Yupapin, "Overlay Fiber Network Based MNRs for P2P Networks," *Procedia Engineering,* vol. 32 , pp. 482-488, 2012.

[82] J. Caubet, O. Esparza, J. L. ,. A. J. Muñoz and J. Mata-Díaz, "Riappa: a robust identity assignment protocol for p2p overlays," *Security and Communication Networks,* vol. 7, no. 12 , pp. 2743-2760, 2014.

[83] A. Srivastava and P. Ahmad, "A probabilistic gossip-based secure protocol for unstructured P2P networks," *Procedia Computer Science,* vol. 78 , pp. 595-602, 2016.

[84] H. Lee and A. Nakao, "Approaches for practical BitTorrent traffic control," in *IEEE Conference on Local Computer Networks*, 2013.

[85] P. Kopiczko, W. Mazurczyk and K. Szczypiorski, "Stegtorrent: a steganographic method for the p2p file sharing service," in *IEEE security and privacy workshops*, 2013.

[86] S. Neuner, M. Schmiedecker and E. R. Weippl, "PeekaTorrent: Leveraging P2P hash values for digital forensics," *Digital Investigation,* vol. 18 , pp. S149-S156., 2016.

[87] Y. Fu, J. Shao, Q. Huang, Q. Zhou, H. Feng, X. Jia, R. Wang and W. Feng, Non-transferable Blockchain-based Identity Authentication, 2023.

[88] S. M R H, M. T. Rahman and N. Mansoor, “Exploration of Hyperledger Besu in Designing Private Blockchain-based Financial Distribution Systems.,” arXiv preprint arXiv:2311.08483., 2023.

Glossary

Confidential Containers: The Cloud Native Computing Foundation (CNCF) has launched a project called Confidential Containers, which aims to enable cloud native confidential computing by utilizing hardware platforms like the Trusted Execution Environment (HW-TEE). The project aims to secure data at the pod-level, removing trust assumptions on the cloud side, and provides resource isolation, data protection, and remote attestation. This approach does not require further modifications to the container image during development.

Root of Trust (RoT): The RoT, as defined by the Global Platform, serves as a computing engine, code, and potentially data, all co-located on the same platform, providing essential security services. It has several types of implementations. More specifically, it can be supported by a Trusted Execution Environment (TEE) or an embedded Secure Element (eSE). Three types of RoT may exist in a trusted platform: i) a RoT for measurement (RTM), ii) a RoT for reporting (RTR) and iii) a RoT for storage (RTS). The baseline for a RoT is to support secure storage (i.e., RTS).

Trusted Execution Environment (TEE): is a secure area within a computer system's hardware or software that provides isolated execution of sensitive code and data. It is designed to ensure the confidentiality and integrity of the executed code and data, even in the presence of potentially compromised or malicious components in the system.

Trusted Computing Base (TCB): refers to the set of all hardware (e.g. memory and storage), firmware, and software components that are critical to enforcing security policies and maintaining the security of a system. TCB often still includes the operating system.

Trust Property: refers to the different elements used by the Level of Trust Assessment component in order to derive to a trust calculation. Among these properties attestation results are included as well as Privacy SLAs, CTI information and other information, as thoroughly reported in D4.1 [3].

Consortium

Space Hellas
www.space.gr

NCSR Demokritos
www.demokritos.gr

Telefonica I&D
www.telefonica.com

RHEA SYSTEM SA
www.rheagroup.com

INESC TEC
www.inesctec.pt

Infili Technologies PC
www.infili.com

UBITECH LTD
www.ubitech.eu

IQUADRAT R&D
www.ucm.es

ICCS
www.iccs.gr

FORSVARETS FORSKNINGSINSTITUTT
www.ffi.no

UNIVERSIDAD COMPLUTENSE DE MADRID
www.ucm.es

INSTITUTO POLITÉCNICO DO PORTO
www.ipp.pt

ERTICO ITS EUROPE
www.ertico.com

Contact Us
privateer-contact@spacemaillist.eu

Co-funded by the European Union
6GSNS

PRIVATEER has received funding from the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme under Grant Agreement No. 101096110