Grant Agreement No.: 952697
Call: H2020-SU-ICT-2018-2020
Topic: SU-ICT-02-2020
Type of action: RIA

D4.5: ASSURED TC-BASED FUNCTIONALITIES

Revision: v.1.0

Work package	WP 4
Task	Task 4.4
Due date	28/02/2022
Deliverable lead	SURREY
Version	1.0
Authors	Nada El Kassem (SURREY)
Reviewers	Liqun Chen (SURREY), Thanassis Giannetsos (UBITECH), Kaitai Liang (TUDE)
Abstract	D4.5 expands on the design of the ASSURED Wallet component, instantiated in each edge device, for supporting both the continuous authentication and authorization of devices, when interacting with the ASSURED ecosystem, but also for the management of all the necessary crypto primitives needed for the establishment of secure and authentic communication channels. Towards this direction, ASSURED is the first of its kind in transforming such a Wallet into a hardware-based trust anchor capable of securely managing Verifiable Credentials with the highest level of assurance. It is based on the integration of a secure element (a Trusted Platform Module (TPM) has been adopted for the instantiaion of all the necessary secure functionalities) and the use of Direct Anonymous Attestation (DAA) for providing verifiable evidence about the presented credentials integrity and origin. Two new standards that have been proposed (by the W3C), namely Verifiable Credentials (VCs) and Verifiable Presentations (VPs), have been adopted for depicting a devices' attributes in a verifiable manner without breaching its privacy as only those attributes that are required for accessing a service or a data bundle will be presented (selective disclosure feature).Based on this novel design, all sequence diagrams and crypto operations that need to be executed by all involved actors (and the underlying TPM) have been fleshed out.
Keywords	Self-Sovereign Identity, Authentication, Authorization, Identity Management, Wallet

Document Revision History

Version	Date	Description of change	List of contributors
v0.1	15.12.2021	ToC	Nada El Kassem (SURREY)
v0.2	07.01.2022	SOTA analysis of the most prominent centralized and de-centralized identity management schemes so as to reason behing the ASSURED decision of adopting the SSI-based technology. (Chapter 2) First draft of the description of the types of credentials managed within ASSURED (Chapter 5)	Kaitai Liang, Shihui Fu (TUDE) Nada El Kassem (SURREY) Stefanos Venios (S5)
v0.3	21.01.2022	First draft of the ASSURED TPM-based Wallet architecture and definition of the required functional specifications. (Chapter 3)	Nada El Kassem, Liqun Chen (SURREY) Edlira Dushku, Benjamin Larsen (DTU) Thanassis Giannetsos, Dimitris Papamartzivanos, Dimitris Karras (UBITECH) Sotiris Kousouris (S5) Meni Onrebach (NVIDIA)
v0.4	28.01.2022	Description of the key management functionalities of the ASSURED TPM-based Wallet and dwelling on the underpinnings of the key hierarchy and key usage features, offered by the TPM, used for authorizing and protecting the use of the AK and the DAA Key. (Chapter 6)	Kaitai Liang, Shihui Fu (TUDE) Nada El Kassem (SURREY) Sotris Kousouris (S5)
v0.5	11.02.2022	Finalization of the ASSURED TPM-based Wallet design including the definition of the swimlanes capturing all the high-level interactions that need to take place between all involved actors (Chapter 3)	Nada El Kassem, Liqun Chen (SURREY) Edlira Dushku, Benjamin Larsen (DTU) Thanassis Giannetsos, Dimitris Papamartzivanos, Dimitris Karras (UBITECH) Meni Onrebach (NVIDIA)
v0.6	18.02.2022	Description of the future research roadmap on the key management functionalities of ASSURED (Chapter 7)	Nada El Kassem, Liqun Chen (SURREY) Edlira Dushku, Benjamin Larsen (DTU) Dimitris Papamartzivanos, Dimitris Karras (UBITECH) Sotiris Kousouris, Stefanos Venios (S5)
v0.7	11.03.2022	Definition of the exact sequence diagrams capturing the interactions between the Wallet and the DAA-Bridge component for enabling the protection and self-issuance of the required verifiable presentations (Chapter 2)	Thanassis Giannetsos (UBITECH) Kaitai Liang, Shihui Fu (TUDE) Nada El Kassem (SURREY) Sotris Kousouris (S5)
v0.9	25.03.2022	Review the document	Liqun Chen (SURREY) Thanassis Giannetsos (UBITECH) Kaitai Liang (TUDE)
V0.95	08.04.2022	Update of the TPM-based sequence diagrams depicting the registration of the TPM-based Wallet (running on the Holder Device) with the Blockchain CA for the issuance of the required Verifiable Credentials so as to correctly add the “Device OK” attribute signed by the DAA Key. Fixed the design error on how to bind a Wallet to a specific device (Chapter 2)	Nada El Kassem, Liqun Chen (SURREY) Edlira Dushku, Benjamin Larsen (DTU) Thanassis Giannetsos, Dimitris Papamartzivanos, Dimitris Karras (UBITECH)
v1.0	15.04.2022	Finalisation of the document	Thanassis Giannetsos (UBITECH), Nada El Kassem (SURREY), Dimitris Karras (UBITECH)

© 2020-2023 ASSURED Consortium

Funded by the Horizon 2020
Framework Programme of the European Union

Editors

Nada El Kassem (SURREY), Liqun Chen (SURREY)

Contributors (ordered according to beneficiary numbers)

Edlira Dushku, Benjamin Larsen (DTU)

Liqun Chen, Nada El Kassem (SURREY)

Kaitai Liang, Shihui Fu (TUDE)

Thanassis Giannetsos, Dimitris Papamartzivanos, Dimitris Karras, George Misiakoulis (UBITECH)

Sotiris Koussouris, Stefanos Venios, Alexandros Tsaloukidis, Konstantinos Charalambous (SUITE5)

Meni Onrebach (NVIDIA)

DISCLAIMER

The information, documentation and figures available in this deliverable are written by the "Future Proofing of ICT Trust Chains: Sustainable Operational Assurance and Verification Remote Guards for Systems-of-Systems Security and Privacy" (ASSURED) project's consortium under EC grant agreement 952697 and do not necessarily reflect the views of the European Commission.

The European Commission is not liable for any use that may be made of the information contained herein.

© 2020-2023 ASSURED Consortium

Funded by the Horizon 2020
Framework Programme of the European Union

COPYRIGHT NOTICE

© 2020 - 2023 ASSURED Consortium

Project co-funded by the European Commission in the H2020 Programme		
Nature of the deliverable:		**R**
Dissemination Level		
PU	Public, fully open, e.g. web	✓
CL	Classified, information as referred to in Commission Decision 2001/844/EC	
CO	Confidential to ASSURED project and Commission Services	

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

DEM: Demonstrator, pilot, prototype, plan designs

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

OTHER: Software, technical diagram, etc.

Funded by the Horizon 2020
Framework Programme of the European Union

Executive Summary

Deliverable D4.5 presents the concept of the **ASSURED TPM-based wallet**, that is used to secure the on- and off-chain interactions by introducing **hardware-based security measures** in the device. Such a trusted TPM-based Wallet design can transform any sw-based Wallet into a hardware-based trust anchor capable of securely managing both Privacy-Preserving Attribute-based Certificates (Privacy-ABCs), as **Verifiable Credentials (VCs)**, as well as all required cryptographic material (keys) with the highest level of assurance. It is based on the use of **Direct Anonymous Attestation (DAA) for providing verifiable evidence about the presented VC's integrity and origin.** The main functionalities for which the Wallet acts as the *trust anchor* are the following: (i) **Creation, management and protected usage of all necessary cryptographic material (keys)**, (ii) **Management of the VCs** (issued by the Blockchain CA during the device registration and enrolment phase) and **Verifiable Presentations (VPs) required for the continuous authentication and authorization (Attribute-based Access Control)** of the device when trying to access a resource (Blockchain service and/or data recorded on the ledger), and (iii) Provision of the required **crypto operations for enabling the creation and of the necessary ABE keys**, binded to specific device attributes, for encrypting the raw system data (traces) prior to been uploaded to the ASSURED Data Storage Engine.

ASSURED adopts a **decentralized digital identity architecture following the current Self-Sovereign Identity (SSI) technologies for enabling secure identity management when interacting with the Blockchain infrastructure for getting access to any of the recorded data.** Two new standards that have been proposed (by the W3C) to realize SSI, namely **Verifiable Credentials (VCs) and Verifiable Presentations (VPs)**, are also been adopted for depicting a devices' attributes in a verifiable manner without breaching its privacy as only those attributes that are required for accessing a service will be presented (*selective disclosure* feature).

In this context, this deliverable provides a detailed description of how this Blockchain wallet will be able to handle the cryptographic operations required to secure the ASSURED Blockchain functionalities; **as it pertains to the device interactions with the distributed ledgers for supporting the auditable security (attestation) policy deployment and attestation execution.** To this end, we first describe the cryptographic material created by the wallet such as the **Attestation Key (AK)** and **Direct Anonymous Attestation (DAA) Key**, which are used to enroll devices in the Blockchain ASSURED services through valid certifications. These keys define two kinds of TPM wallet certifications:

- The first certification is created during registration with the **Privacy Certification Authority (Privacy CA)** that issues credentials on the TPM wallet Attestation Key (AK), which will be used later to create assurance claims about their states. This registration certification **binds a specific TPM Wallet to the holding device** so that it can be the only entity that can use the credentials and verifiable proofs needed for accessing a service or data resource; *thus, enabling the provision of publicly verifiable evidence that the presented set of attributes and identifiers really belong to the claimed device*.

- After activating the AK credentials, the TPM wallet creates and certifies the DAA key that is connected to the AK in the key hierarchy. Next, the AK will be used for signing the traces measured by an authorized tracer while the DAA key is used to create **Verifiable Presentation (VPs)** based on its VCs, which include all the necessary and verified attributes for getting access to all of the offered services. The use of the loaded Blockchain VCs for creating appropriate verifiable presentations, signed under the DAA Key, is also protected

through **policy-based authorization**, that ensures that a device is in a correct configuration state.

Furthermore, the cornerstone of **key management** is also elaborated, which is one of the key functionalities that is supported by the ASSURED TPM-based Wallet when it comes to the **creation, usage and storage of various cryptographic material** needed for supporting the secure participation of a device in the target "Systems-of-Systems" environment. A reliable key management should establish and specify rules to protect its **confidentiality**, **integrity**, **availability**, and **source authentication**. In this direction, the TPM wallet functionality is used to control and authorize access to private or public ledger channels based on the key polices and attributes of the TPM wallets, which were authorized by the SCB when enrolling the device.

Different types of keys need to be stored in the device, such as **Attestation Keys**, **encryption keys** and **Blockchain keys**, which are used for supporting different functionalities provided by the ASSURED framework. For example, a TPM is identified by its **Endorsement Key**, while it uses an Attestation Key to provide attestation services such as signing a set of platform configuration registers and providing a timestamp, while the DAA key is also required to perform attestations while preserving the device's privacy. A Blockchain key is used to access the ASSURED Blockchain services, and to protect the communications between all edge devices with the backend infrastructure.

Overall, ASSURED adopts the concept of SSI and Privacy-ABCs for enabling privacy-preserving identity management for all devices trying to interact with each other or the backend Blockchain infrastructure. In this context, ASSURED extends the ESSIF-Framework by building a new component on the Holder side that **enables the use of hardware-based keys and offers the possibility to bind VCs to the Wallet of the Holder**. In this way, we transfer the root of trust of the SSI ecosystem purely on the digital wallet by considering an underlying TPM as part of the wallet; this, however, can also be extended to any other type of secure element, without making any assumptions on the trustworthiness of the other layers. Besides the novel design of the ASSURED TPM-based Wallet, D4.5 also captures the exact sequence diagrams and crypto operations that need to be executed between all involved actors (and the underlying TPM) for enabling the aforementioned functionalities of key management and continuous authentication and authorization. The full implementation of the TPM wallet functionalities will be released in the next version of this deliverable.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Trust is a critical component of any data sharing and identity management system and as such is also crucial in the context of ASSURED for enabling both the **continuous authentication and authorization** of devices, when interacting with the ASSURED ecosystem (especially the Blockchain infrastructure for getting access to *operational- and attestation-related data*) but also for the **management of all the necessary crypto primitives** for the establishment of secure and authentic communication channels. Recall that, as described in D3.1 [26], key management lies at the heart of ASSURED leveraging the key hierarchy and key usage functionalities - of the underlying Trusted Component (a Trusted Platform Module (TPM) in all of our instantiated scenarios) - for managing the different types of keys leveraged: (i) an **Attestation Key** for safeguarding the integrity of the *traces* produced by a Prover device [32, 46], (ii) a **DAA key** for enabling the privacy-preserving (when needed) device authentication and sharing of operational data in an anonymous and unlinkable way (leveraging short-term anonymous credentials - pseudonyms) [34, 47], (iii) **Symmetric keys** for safeguarding the confidentiality of device communication [26], and (iv) **Public-private key pair** for the authentication and interaction of a device with the Blockchain infrastructure [31].

In this context, ASSURED is employing strong cryptographic protocols for **increasing trust without letting any privacy violations** be technically possible. More specifically, a novel **decentralized Attribute-based Encryption (ABE) scheme** has been designed that enables devices to encrypt the data, to be shared through the Blockchain, leveraging keys that they create locally (through their attached TPM) and their usage is protected by specific attributes [34]. *This means that the requesting parties in order to be able to decrypt the data, they need to have access to the required types of attributes so that their TPM can correctly create the necessary decryption keys*. While such an approach brings the **benefit of shifting trust to the "edge" and give more control on the device side, it also creates an interesting discussion on the acceptance factors and the needs when it comes to attribute-based identity management**.

Over the past years, a number of technologies have been developed to build **Privacy Preserving Attribute-based Credentials (Privacy-ABCs)** in a way that they can be trusted, like normal cryptographic certificates, while at the same time they protect the privacy of their Holder [56]. Such Privacy-ABCs are issued just like ordinary cryptographic credentials using a digital secret signature key. However, Privacy-ABCs allow their Holder to transform them into a new token, in such a way that the privacy of the user is protected. As described in D4.1 [27], in ASSURED, such **Attribute-based Access Control (ABAC) mechanisms are employed for verifying the digital identity of a device prior to be granted access to the recorded attestation- and operational-related data**.

Digital identity, of a device, is the set of all characteristics (which we call **"attributes"**) that have

been attributed to this entity within a specific scope [43]: e.g., *getting access to the assurance claims (attestation data) that have been recorded for all sensors and IoT Gateways of a specific manufacturing floor (as part of the* Smart Manufacturing *use case* [29]). This can be interpreted to device system attributes that need to be depicted to the Security Context Broker (SCB) when querying for specific data and are managed by the attached TPM that then uses them for "unlocking" the generation of the appropriate ABE decryption keys. It is important to note that there are different aspects of a device's identity, which can be used in different contexts and situations. These different aspects are called "*partial identifiers*" [1,25]. **Depending upon the situation and the context, a device may be represented by a different set of partial identifiers.** Such identifiers can embody various information such as OS version, type of libraries installed, type of CPU micro-controller, etc. - for verifying the **correct configuration of the device** - to **security claims on the correct execution of specific software, that can be outputted from the ASSURED attestation enablers.**

The endmost goal is to enable the devices to provide (in a verifiable manner) all the required attributes and identifiers needed for attesting to their level of trustworthiness as required for securely interacting and getting access to the various ASSURED services and the Blockchain infrastructure. However, one of the main challenges in such settings is how to ensure that the device is in fact the one it claims to be. **How can someone be sure that the presented set of identifiers and attributes really belong to the claimed device?**

Towards this direction, ASSURED adopts a **decentralized digital identity architecture following the current Self-Sovereign Identity (SSI) technologies for enabling secure identity management when interacting with the Blockchain infrastructure for getting access to any of the recorded data.** Two new standards that have been proposed (by the W3C) to realize SSI, namely **Verifiable Credentials (VCs) and Verifiable Presentations (VPs)**, are also been adopted for depicting a devices' attributes in a verifiable manner without breaching its privacy as only those attributes that are required for accessing a service or a data bundle will be presented (*selective disclosure* feature). **The management and (self-) issuance of all such VCs and VPs is handled by the ASSURED Wallet which is further protected through the use of hardware-based keys supported by the underlying TPM.** This TPM-based Wallet is responsible for both enabling a high level in the differential credential security model (by anchoring DAA assurance claims to the generated VPs) but also for securely managing all of the crypto primitives, tokens and keys needed for safeguarding the integrity, authenticity and confidentiality of all shared data.

1.1 Creating Trust in Data Management Transactions

DLT systems rely on trust anchors of different types, each being important in terms of some dimension of policy, technology, data, security, assurance and more. For instance, in ASSURED, trustworthy and secure data interoperability is a crucial factor for both operational- and attestation-related data when shared to other devices and stakholders: Only authenticated and authorized entities should be able to be granted access to such data so that privacy-preservation is guaranteed. This forms the corresponding obligations of **data access rights** ("who, when and where" can access data records and can be used for which purpose) and **data exchange and sharing in a secure and validated way** (i.e., enabling the seamless sharing of data across multiple networks without loss of information).

Towards this direction, **TPM-based Wallets** are one of the core building blocks of ASSURED, and form the basis for enhanced **security**, **privacy** and **reliability** guarantees for continuous authentication and authorization. ASSURED TPM wallets communicate with the rest of the Blockchain

components to perform secure (attribute-based) access control to the distributed ledgers. Thus, the use of such wallets is to enable **secure storage of cryptographic key material** and **platform measurements** that ensure that the connected devices remain at the required level of trustworthiness. The wallet will also interface with the rest of the Blockchain-enabled components of the ASSURED ecosystem to perform secure on- and off-chain data management functionalities, and for recording the attestation evidence to the ledgers through communicating with the Blockchain Peers [27]. Such a communication will only be allowed through the TPM-based Wallet after providing the necessary VPs granting access to the device to access the Blockchain service. Based on this, the TPM wallet will undertake the basic function of storing the keys and the trusted wallet may offer the functionality of encrypting information to ensure secure communication channels between the devices and peers.

Blockchain wallets provide a *secure environment*, that constitutes an isolated, secure execution environment for **protecting device data through cryptographic algorithms such as hash functions, symmetric and asymmetric encryption/decryption and digital signatures.** The TPM-based Wallet is required by ASSURED use cases to enable **continuous authentication and authorization as well as remote security attestation and enhance the confidentiality and integrity of the exchanged data.** Such as trusted wallet architecture also allows for the safe deployment and verification of software updates, as needed by the *Smart Aerospace* use case: *Only devices with a specific OS version (as an attribute) are entitled to securely (through the Blockchain Public-Private key pair) download, install and execute a specific software update and should be able to verify their configuration update through the issuance of an updated VP*.

Taking into consideration the above, it follows that TPM-based wallets provide functionalities that are essential towards the implementation of the security services provided by ASSURED. Specifically, as it pertains to the offered attestation services for verifying the operational assurance of a device while preserving the privacy of this device (against, for instance, implementation disclosure attacks) and achieving different levels of trustworthiness depicted by different sets of attributes and partial identifiers. Recall that, as described in D3.6 [33], device system properties that are included as attributes in a VP should not leak any information about the state or the software running in a device since this can be used by an adversary to possibly identify system vulnerabilities to be exploited [37]. Thus, only the attributes required for accessing a service (**selective disclosure**) should be released as part of an issued VP in **zero-knowledge manner**[1]. To this end, ASSURED aims to manage the provided **Blockchain-control services** through the specification of novel **TPM-based security** and **privacy-preserving protocols**, for advancing the state-of-the-art in scalability and computational efficiency, as well as the enforcement of **security (attestation) policies** through **smart contracts**, which are downloaded through the TPM wallet.

For the latter, the **attestation enablers**, besides their usage towards verifying the operational assurance of the target SoS, also **output the security claims that can be used as VC attributes**. For instance, one device might need to attest to maintaining specific configuration properties prior to getting access to the Blockchain and recorded data. **This is achieved through the TPM-based Wallet that acts as the "*bridge*" for the secure authentication and authorization of the device, the secure download of the attestation contracts and the appropriate recording and verification of these security claims as attestation outputs.** In order to guarantee that only trusted and uncompromised devices can participate in the envisioned supply chain ecosystem, all involved devices will use their TPM wallets to secure the boot mechanism, and their trust level will be continuously attested and assessed. To this end, the TPM wallets create an attes-

[1] We have to note here that the ASSURED CFA will be enhanced with zero-knowledge capabilities in the second release of the attestation enablers scheduled for M30.

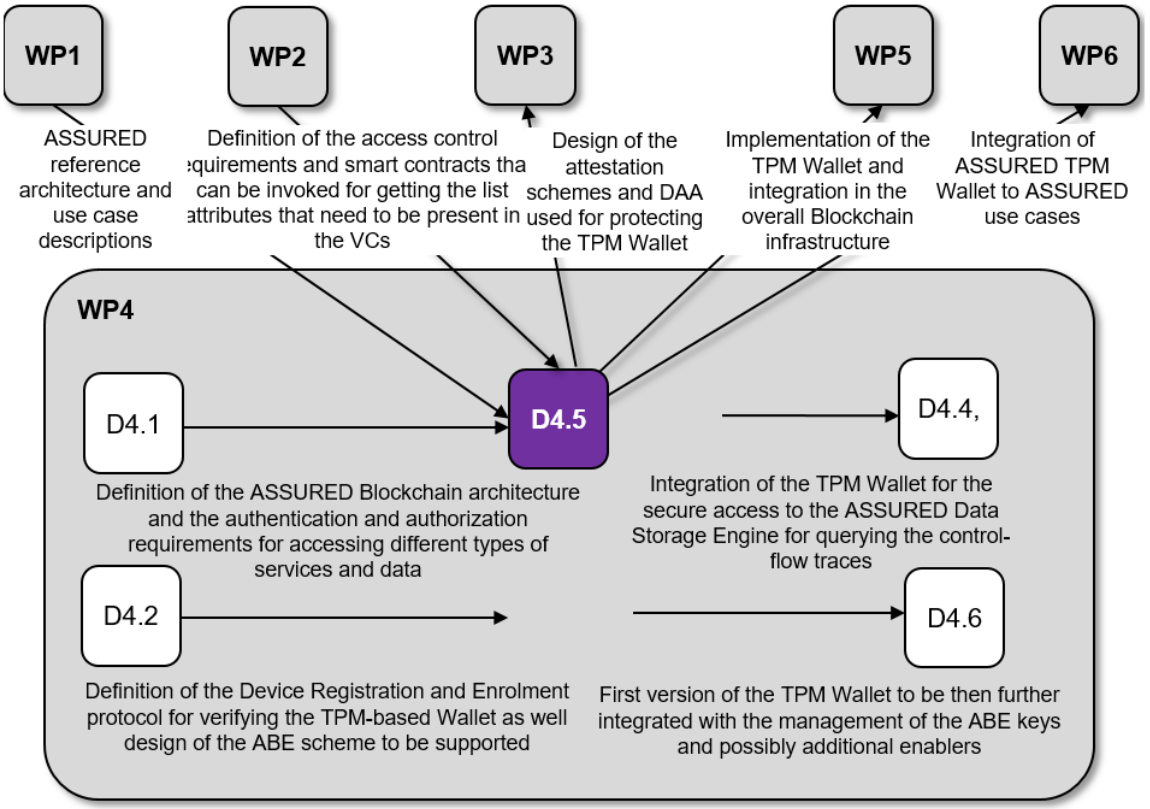

Figure 1.1: Relation of D4.5 with other WPs and Deliverables

tation evidence that includes the respective platform's **integrity state** (which is the hash value held by the device's PCRs at the end of the secure boot process in case of static attestation or authorized signed traces in Contol Flow Attesation CFA), which will allow any other party to check whether the data stems or was acknowledged by a trusted entity.

Recall that attestation evidence is a form of **signature over a software/firmware measurement in a PCR using an Attestation Key (AK) protected by the TPM Wallet**. This is done for achieving **data sovereignty**, in the context of *data provenance* and *data integrity*, so that the Verifier can make sure of the correctness and authenticity of the received (control-flow or configuration) traces prior to forwarding them to the appropriate attestation enabler for verification. Such signatures, enhanced also with a signature originating from the Verifier device, are also accompanying the attestation reports recorded on the ledger within the corresponding obligation of data exchange and sharing in a validated way. Essentially, so that any external stakeholder can verify the sovereignty of an attestation record highlighting the operational status of the respective device. To create such signatures, the TPM Wallet must have a valid AK credential (Cred), which is a certificate containing the public AK and a proof that the AK originates from a genuine TPM. This credential is provided by a trusted party known as the **Privacy CA** during the Device Registration and Enrollment phase [34] which (among other things) also certifies the presence of a valid TPM Wallet and the subsequent creation of all the necessary cryptographic material (i.e., keys) and the appropriate authorizing policies that regulate the usage of the created keys (e.g., *AK cannot be used for signing anything else than traces originating from an authentic Tracer* [26]). This credential also includes the set of attributes that characterize the status of this device needed prior to getting access to any of the Blockchain services.

This credential, along with a *token* [34], is then forwarded to the Blockchain Certification Authority (CA) that is responsible for authenticating the device identity and its issued attributes, by verifying the provided credential. However, as will be described in Chapter 3, in order for the Blockchain CA to ensure that the device is the intended holder of this $Cred$ (*ensure that the device is in fact the one it claims to be*) it also **requests evidence that bind the TPM Wallet (holding this $Cred$) to this specific device**. This is achieved by the use of Direct Anonymous Attestation (DAA) for providing verifiable evidence about the presented credentials' integrity and origin. Essentially, the devices' Wallet is enhanced with a **DAA-Bridge** component (forming the **ASSURED TPM Wallet**) that allows the use of credentials only for the binded device and only when the device is at a correct state. *ASSURED, in this context, is the first project of its kind that brings together the two worlds of SSI and trusted Computing for transforming a Wallet into a hardware-based trust anchor capable of securely managing credentials*.

The Blockchain CA then issues a **Verifiable Credential (VC)** linked to the valid attributes required for accessing specific services or stored data. This VC, alongside the Public-Private Key Pair needed by the device for securely interacting with the Blockchain, is securely sent and stored in the device's TPM Wallet. Whenever the device wishes then to access the Blockchain infrastructure, it first requires the necessary list of attributes that need to be exhibited (through already deployed smart contracts) and then issues a Verifiable Presentation (signed by the DAA Key) *selectively disclosing* only those required attributes. **Since the VP is signed by the DAA-Bridge, the Blockchain Security Context Broker (SCB) is sure of the correctness of the device holder (authenticity) and that the respective key has not been compromised.**

1.2 Relation to other WPs and Deliverables

In what follows, Figure 1.1 depicts the relationship of Deliverable D4.5 with other Work Packages (WPs) as well as the other tasks in the same WP(4). The main purpose of this document to design the ASSURED TPM-based Wallet for supporting the secure on- and off-chain data management and data sharing behaviours defined in D1.4 [28] for the envisioned use cases. Thus, D4.5 defines the novel TPM-based Wallet, protected through a hardware-based key (DAA Key), and all of its functionalities when it comes to **continuous authorization and authentication, identity management, and secure management of all cryptographic material**. These functionalities support the advanced crypto primitives defined in D4.2 [34] including symmetric encryption, attribute-based encryption, hash message authentication code and digital signatures for guaranteeing the compliance to the data confidentiality, data integrity and data traceability models defined in WP1 and WP3.

The use of the TPM-based Wallet is to maintain security, privacy and trustworthiness guarantees throughout the entire lifecycle of a device by employing advanced remote attestation mechanisms defined in D3.2 [32] and D3.6 [33], as a central building block for the trusted exchange of data as well as for secure device management. In this direction, the TPM-based Wallet will run the crypto protocols to provide secure device Blockchain enrollment as explained D4.2 [34], ensure data integrity and verification of the attestation data as defined in WP3, run Attribute-based Access Control (ABAC) and Searchable Encryption (SE) schemes (defined in D4.3 [31]) for accessing the ASSURED ledgers.

The outcome of Deliverable D4.5 is to provide a detailed explanation on the TPM-based Wallet and all of the TPM-enabled functionalities, for interacting with the Blockchain, required for the secure data management operations: starting from secure enrollment, to creating attestation keys

and accessing the Blockchain ledgers relying on appropriate type of the cryptographic materials needed to perform the tasks in a secure and safe manner. It is intended to support D4.6 which designs the final version of the ASSURED TPM-based Wallet considering all of the functional specifications for supporting the correct execution of the ABE scheme as well as for managing VCs aligned with the current W3C data model. WP4. Last but not least, WP5 that undertakes the development of the integrated framework and WP6 that aims to its validation in the context of the pilots, inherit the baseline of the TPm Wallet and the high-level interactions that need to be validated.

1.3 Deliverable Structure

This deliverable is structured as follows: **Chapter 2** gives an overview of the most prominent identity management schemes that have been proposed over the years covering both extremes of purely centralized vs. decentralized mechanisms. It highlights the benefits and challenges of each approach so as to reason over the ASSURED decision to proceed with the adoption of a decentralized digital identity management architecture following the current SSI technology. This is further elaborated in **Chapter 3**, where the focus is on describing the novel design of the ASSURED TPM-based Wallet capable of providing the root-of-trust for the secure on- and off-chain interactions between the devices and the other ASSURED components. This Wallet enables the use of hardware-based keys and offers the possibility to bind Verifiable Credentials (VCs) to the Wallet of the Holder. In this way, we transfer the root of trust of purely on the digital wallet by leveraging also crypto operations of the DAA (ECC, Blind Signatures, Zero Knowledge Proofs) towards providing additional trust assurances. After having defined the detailed architecture of the ASSURED TPM-based Wallet, **Chapter 4** proceeds with a description of the security functionalities and services that rely on the integration of the Wallet and its interaction with the other components in the overall ASSURED ecosystem - covering both authentication and authorization as well the secure interaction with the ledger for reading/querying attestation policies and recording attestation results. All of these services require the secure management of the device's credentials prior to getting access to the ASSURED ledger. Types of credentials managed within ASSURED are documented in **Chapter 5** dwelling into the details of how they are issued during the device registration and enrollment phase but also how they are then used (throughout the device's lifecycle) for getting access to a specific service or data resource (Attribute-based Access Control). **Chapter 6** then proceeds with the description of the functional specifications of the Wallet regarding the second core functionality supported as it pertains to the management of all the necessary cryptographic material (keys) for the secure communication and confidentiality and integrity guarantees required for safeguarding the execution of the attestation enablers. **Chapter 7** puts forth some of the future research questions to be addressed in the second release of this deliverable related to the establishment of anonymous secure channels and the better storage of all required keys (i.e., Attestation Key, DAA Key, Symmetric Keys, Blockchain Keys) as part of the key hierarchy of the underlying trusted component. Finally, **Chapter 8** concludes this deliverable.

Chapter 2

Centralized & Decentralized Digital Identity Schemes

As aforementioned, one of the core functionalities offered by the ASSURED TPM-based Wallet is the **continuous authentication and authorization of the devices when interacting with the Blockchain infrastructure** for either querying operational- or attestation-related data or recording attestation results (in case a device is acting as a Verifier for the execution of an attestation policy). *The second functionality is the management of all cryptographic material needed for the secure device pariticipation in the ASSURED ecosystem, as described in Chapter 6*.

In this context, identity management is achieved through the use of Verifiable Presentations (based on the VCs that were issued by the Blockchain CA) that can disclose evidence about only those device attributes that are needed for accessing a service. **Such attributes can embody various information including OS version, type of libraries installed, type of CPU micro-controller, etc. - for verifying the correct configuration of the device - as well as security claims on the correct execution of specific software, that can be outputted from the ASSURED attestation enablers.**

The endmost goal in ASSURED is to enable the devices to manage such VPs without the need to always rely on (trusted) centralized entities (such as the Blockchain CA) for issuing them. Trusted Issuers can verify the attributes of the devices for issuing the required Verifiable Credentials (during registration) but the devices should then be able to locally and securely self-issue presentations that can be verified by any other entity. **This is aligned with the main vision of operating in a Zero Trust ecosystem but also allows for enhanced scalability since trust is shifted to the devices minimizing the issuance interactions with the backend identity management system:** In ASSURED, we can distinguish three parties: devices, issuers and verifiers. Issuers are the source of credentials and usually are trusted organizations and entities (Blockchain CA). However, devices can also be issuers since, for example a Program Logic Controller (PLC) sensor (in the context of the Smart Manufacturing use case [30]) can issue a digitally signed credential about the integrity worker location measurements.

In what follows, we provide a documentation of the most prominent identity management schemes (considering both centralized and de-centralized approaches) as well as the employed cryptographic primitives used for continuous authentication and authorization. This culminates with the reasoning behind ASSURED adoption of a **decentralized digital identity architecture following the current Self-Sovereign Identity (SSI) technologies for enabling secure identity management when interacting with the Blockchain infrastructure for getting access to any of the recorded data**. The SSI model is the core underlying technology behind the novel design

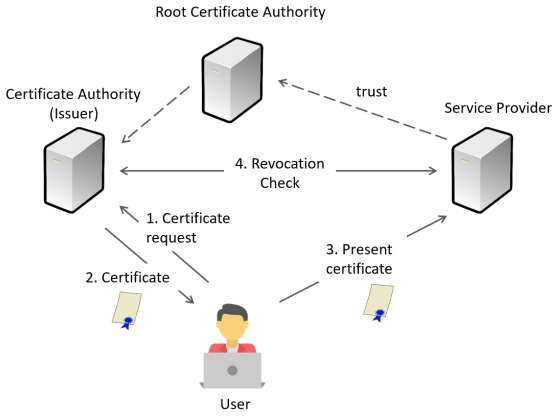

Figure 2.1: Entities in an Privacy-ABC system and their interactions

of the ASSURED TPM Wallet (Chapter 3) for transforming a **(SSI) Wallet into a hardware-based trust anchor capable of securely managing Verifiable Credentials with the highest level of assurance**. It is based on the use of Direct Anonymous Attestation (DAA) for providing verifiable evidence about the presented VC's integrity and origin.

2.1 Centralized Identity

One of the currently used ways of doing online identity management is to use a standard digital signature scheme in a Public Key Infrastructure (PKI). Probably the best-known example is the SSL/TLS PKI, which uses X.509 certificates [5] to provide secure connections in browsers using the HTTPS protocol. This system requires that all certificates be obtained from a relatively small list of trusted authorities, called Certificate Authority (CA), and that any changes to these certificates also be approved by someone in this chain of trust.

Figure 2.1 shows the steps of the protocol in more detail. The device starts by generating a key pair, and sends a certificate request that includes the generated public key to the CA. The CA creates, signs and returns the certificate to the device which stores it along with the associated private key. To authenticate to a Service Provider, the device uses the certificate's private key to sign a Service Provider-specific challenge. The service provider verifies the signature and validates the certificate. This involves verifying the CA signature in the certificate, making sure that the CA is a trusted issuer and making sure that the certificate has not expired and is not revoked.

Public key infrastructure (PKI) implements this centralized trust model by inserting reliance on a hierarchy of certificate authorities. These certificate authorities establish the authenticity of the binding between a public key and its owner via the issuance of digital certificates. Therefore, the PKI trust model is centralized towards the certificate authorities as trusted third parties.

2.2 Federated Identity

In federated identity systems [19], an IdP stores all the attributes associated with a device and vouches for the identity of the devices. When a device wants to access a service of a Service Provider (SP), the SP delegates the authentication to the Identity Provider (IdP). So, for identity

federation to take place, the SP must trust the the authentication assertions of the IdP. Trust relationships are usually established by a set of contracts defining obligations and rights each party has and policies each member has to follow. This results in is a Circle of Trust, in which each partner trusts on the assertions made by another partner. Federated identity is the means by which Web applications can offer devices cross-domain single sign-on (SSO), which lets them authenticate once and thereafter gain access to protected resources and Web sites elsewhere. The two most widespread federated identity protocol families are: (1) SAML for large-scale federations such as eIDs and (2) OAuth for web authorization in combination with OpenID Connect for web authentication [50].

However, federated identity management has introduced new and increased security and privacy risks, since in this setup identity is shared across domains. More specifically, it is usually the case that when the certificates contain device-specific attributes they also require disclosure of all these attributes with every usage, which violates the privacy paradigm of data minimization (even though it can be avoided with Online IdPs or Single Sign-On (SSO) schemes). A common weakness through all solutions is the problem of linkability, meaning that the IdP typically learns which Service Provider the device is trying to access. This information is often provided to the IdP in order to protect the security token or to redirect the device to the right location. A related problem to linkability, is the problem of traceability, meaning that we now have a single party, the IdP, which will learn about every service provider that the device wishes to access. At the same time, any two service providers that compare logs will also be able to identify if the same device has been using both services.

2.3 Device-Centric Identity

In 2008, Kim Cameron, Reinhard Posch and Kai Rannenberg published their work on “A device-Centric Identity Metasystem”, describing an abstracted design for a system which puts the device in control of their own data, the accumulation of that data, and its release to third parties [18]. In this paper, the authors make it clear that the core requirement for device control is that the device is placed in-between the IdP and the SP and any information flow goes through the device so that he can maintain control of his information. Devices are storing credentials locally and then they control what they share with others, by presenting “claims”.

Modern cryptography has offered tools to realize device-centric identity models. The basis of these tools has been laid by pioneer David Chaum, who put forth the principles of anonymous credentials [22], group signatures [21], and electronic cash [20]. The cryptographic research community has since developed and extended these ideas. The first private credential system that is practical and yet provably secure was presented by Camenisch and Lysyanskaya [14]. That first scheme is based in the RSA assumption. A simplified version of it, called Direct Anonymous Attestation [11], was standardized by the Trusted Computing Group in 2004 [59]. Subsequently, schemes based on elliptic curve cryptography have been put forth and protocols providing an extended set of features were developed.

After early theoretical development, two competing realizations of anonymous credentials emerged, namely Idemix [17] and U-Prove [53]. During 2010-2015, an EU-funded research project called Attribute-based Credentials for Trust (ABC4Trust) came to present a unified architecture and delivered an open reference implementations of anonymous credential systems [54]. It also introduced the more correct term: Privacy Attribute-Based Credentials (Privacy-ABCs).

Privacy-ABCs follow the same “offline” approach as X.509 certificates, i.e., devices receive a

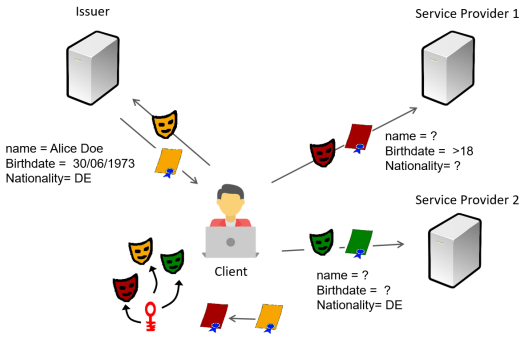

Figure 2.2: Entities in a Privacy-ABC system and their interactions

credential from a trusted issuer. The credential contains a set of certified device attributes that the issuer vouches for, e.g., the device's name, age or address, and that can be used to convince a service provider of the validity of such claimed attributes. Unlike classical X.509 certificates, which have to be disclosed entirely and expose a static value whenever used, these Privacy-ABCs credentials allow the device to derive dedicated one-time use tokens that reveal only the information that is minimally necessary.

With the device in the middle, there is no direct contact between IdP and SP, resolving the linkability problem inherent to the centralized models. Furthermore, claims are in fact anonymous, meaning that the device doesn't transfer them to the RP directly. Instead, the statement in the claim is proven to the RP in a zero-knowledge fashion. This further protects the device's privacy, because it makes the device unlinkable between two interactions with a relying party, which resolves the traceability problem as well.

More precisely, when the device wants to access a service, the service provider first responds with a presentation policy, stating the requirements the device has to fulfill, e.g. requests for proofs of certain attributes. Let's say one service provider might request devices to be older than 18 years or another service provider might have a policy that devices should live in a European country (see Figure 2.2). If the device's credential can satisfy the policy, it derives a so-called presentation token from the credential. The presentation token contains only the minimal amount of attributes requested by the policy. Importantly, the device can derive only presentation tokens that are consistent with the information certified in the credential, and the service provider can verify the token against the policy and be convinced about its correctness.

2.3.1 Properties of Privacy-ABCs

To summarize, Privacy-ABCs have the following properties with respect to privacy protection as also adopted in ASSURED:

- **Minimal Disclosure:** Privacy-ABCs allow the devices to derive authentic and verifiable tokens from their credentials that contain only the required information, therefore avoiding disclosure of all the attributes in the credentials.
- **Unlinkability:** Another key feature of Privacy-ABCs is unlinkability that comes in two different types, namely, unlinkability to their issuance, and unlinkability across different presentations. The IdP is only involved once, at credential issuance, but does not need to be contacted at every authentication request. This means that the issuance and the usage (presentation) of a credential cannot be correlated.

- **Partial Identities and identifiers:** The concept of Pseudonyms in Privacy-ABCs enable devices to generate an unlimited number of unlinkable pseudonyms from their secret key and establish different profiles across different services.

2.3.2 Adoption of Privacy-ABCs in ASSURED

Privacy-ABCs, as much attractive properties as they have, they are only now start becoming the norm for enhanced privacy-protection in various application domains [42]. *One of the main reasons is concerns regarding their applicability*. **This argument stems from the fact that Privacy-ABCs require devices to securely manage their credentials and key material. It is important that the devices keep this material very well protected, for example by keeping them on trusted hardware tokens, such as TPMs and be responsible for the backup and recovery in case of loss.**

However, adoption of a technology by devices is a more complicated concept. In reality, acceptance of a technology is tightly connected to a cost-benefit assessment of the technology. On one side, the benefits of Privacy-ABCs can be expressed in terms of their perceived usefulness for protecting privacy. On the other side, the costs of the Privacy-ABCs are mostly connected with usability issues. We can study this more formally by coming up with a Technology Acceptance Model (TAM) for PETs. During the ABC4Trust project, Benenson et al. applied an adapted version of TAM model and made an empirical evaluation of the factors that influence user acceptance of Privacy-ABCs during the two user trials of the project [8]. The study found that the benefits outweigh the costs. Devices thought that usability costs can be compensated by other, more usable features. So there is a positive cost-benefit assessment of the Privacy-ABC technology. Service providers also engage in a cost-benefit trade-off involving many factors related not only to internal processes and business models but also to the external environment, such as legislation, user demand or global infrastructure readiness [45].

Hence, the reasoning behind the adoption of such privacy-preserving ABCs in the context of ASSURED for both enabling the privacy-preserving device and user authentication, when needed, but also for the privacy protection of any sensitive operational data shared within the ASSURED ecosystem. Especially, in the context of the Smart Cities use case where the focus is on public safety based on monitoring the movement of suspicious people and/or material, it is of paramount importance to be aligned with the privacy protection regulations. **ASSURED provides an enhanced TPM-based Wallet that can manage the issuance and use of such priacy-preserving ABCs, in an agnostic way to the users/devices, thus, alleviating any usability concerns while at the same time provide flexibility on the type of attributes and partical identifies that can be considered as part of such verifiable credentials. In this context, ASSURED enhanced the current W3C data model.**

2.4 Decentralized Digital Identity Infrastructure

Besides the use of privacy-preserving ABCs, another important decision behind the use of advanced Wallet concepts in ASSURED, is the adoption of decentralized digital identity architecture following the current Self-Sovereign Identity (SSI) technologies for enabling secure identity management when interacting with the Blockchain infrastructure for getting access to any of the recorded data. Self-sovereign identity technologies have emerged to increase the trust in digital identity and personal data across data transactions (**data and digital sovereignty**). Even

though Self-sovereign Identity (SSI) is a term that is not always used consistently, a few key properties of the concept have emerged. These can be traced back to the so called Ten Principles of Self-sovereign Identity proposed by that apply a strong user focus to identity management.

No single entity acts as a central authority that has control over identifier origination and/or credential issuance. Participants manage their own identifiers and credentials without requiring any permissions. A Blockchain is used to support the mapping of keys to identifiers by acting as an integrity-protected "bulletin board" for public key infrastructure (PKI) [48]. Two new standards are being proposed to realize SSI, namely, Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs) [60]. Both are currently being developed by the World Wide Web Consortia (W3C).

The main components of SSI systems are the decentralized identifiers (DIDs) [61]. The DID is a permanent, universally unique identifier and cannot be taken away from its owner who owns the associated private key. DIDs are typically used together with Verifiable Credentials (VCs) [60] and verifiable claims that enable making any number of attestations about a DID subject. Verifiable Credentials employs cryptography to enable tamper-proof and digitally signed claims, which can be selectively revealed to the service provider and can be verified without contacting a centralized trust source. Camenisch-Lysyanskaya Zero-Knowledge Proofs [15] has been recommended as one of the standard cryptographic proofs for Verifiable Credentials and more recently, a draft for the W3C Recommendation utilizes BBS+ signatures [7] to provide the capability of zero knowledge proof disclosures [49].

In SSI ecosystems, there are three key roles: issuers, holders and verifiers. At this point it is important to clarify that the use of Blockchain is not necessary for the operation of an SSI system. It is one option that offers the storage, update, deletion / recall functions of DID that are required, but this can also be achieved in a more decentralized manner through the use of secure elements at the device level for providing evidence on the correct execution of such a function (current ASSURED concept). In this context, the level and type of sharing of data remains on the device control.

2.4.1 Adoption of SSI Technologies in ASSURED

Support for SSI is growing, especially in Europe and North America. There are a few notable examples [4]. The EU has adopted the SSI vision by including the "The European Self-Sovereign Identity Framewor" (ESSIF) [39] as part of the "European Blockchain Services Infrastructure" (EBSI). EBSI is an EU-initiative aimed at providing a public Blockchain infrastructure for government services (at EU level). ESSIF, in turn, builds on this infrastructure to develop and promote SSI solutions. In addition, the system that will emerge under the ESSIF must be compatible with GDPR and eIDAS legislation in order to offer the highest possible guarantees at the level of protection of European citizens. The goal is to provide a user and device-based Blockchain-based identity management system, which, like a real wallet, will store different identifiers and attributes in a digital wallet, which will be under the full control of the users/devices.

This is exactly what ASSURED strives to offer. It designs a novel Digital Wallet based on the use of TPMs, as the underlying root-of-trust, for establishing trust across multiple stakeholders and devices when accessing the Blockchain infrastructure for data sharing. Such a design can also be adopted in the SSI ecosystem for better safeuarding the role of the Holder (user/device) that now has complete control of its verifiable credentials and proofs over them (Chapter 3).

At the same time, the European Commission proposed the update of EU Regulation 910/2014 (eIDAS) for the creation of a European Digital Identity Framework. This proposal is currently being

discussed by European Parliament committees and includes several aspects of SSI. In particular, it supports the idea of a ”European Digital Identity Wallet”, which is a product and service that allows the user to store credentials and features related to their identity and display them online and offline upon request. in order to have access to services. However, the proposal still lacks clarity on some security and privacy measures, which are still under discussion.

Finally several open-source communities, standard-setting organizations and non-profit organizations aim to define, standardize and provide tools for the SSI ecosystem. With respect to standardization, the most prevalent components of SSI technology are proposed by the World Wide Web Consortium (W3C) [60] and the Decentralized Identity Foundation (DIF) [38].

Chapter 3

Architectural Overview of the TPM Wallet

As aforementioned, ASSURED adopts the concept of SSI and Privacy-ABCs for enabling privacy-preserving identity management for all devices trying to interact with each other or the backend Blockchain infrastructure. In this context, ASSURED extends the ESSIF-Framework by building a new component on the Holder side that enables the use of hardware-based keys and offers the possibility to bind Verifiable Credentials (VCs) to the Wallet of the Holder. In this way, we transfer the **root of trust of the SSI ecosystem** purely on the digital wallet by considering an underlying TPM as part of the wallet (this, however, can also be extended to any other type of secure element that offers the technical requirements as listed in D1.1 [30]), without making any assumptions on the trustworthiness of the other layers.

ASSURED has implemented various design schemes towards performing **secure and lightweight on-chain interactions**. Specifically, these interactions may include **reading data**, **reading attestation policies**, **recording data**, **recording attestation results** or **performing various queries**. Therefore, in order to enable secure interaction of the devices with the ledger, as well as to accomplish the **data-sharing agreements and behaviors** that were defined in D1.4 [28], we construct such a novel **hardware-based wallet**, which will enable the management of all the necessary cryptographic materials that support all the aforementioned functionalities.

The concept of **Self-Sovereign Identities (SSI)** involves the management of digital identities for users or devices in a decentralized manner. These identities are depicted through the use of **Verifiable Credentials (VCs)**, that serve to prove the identity of a device when communicating with another target device or component, as well as **Decentralized Identifiers (DIDs)**, which are issued by a decentralized platform and act as a proof of ownership or digital identity. The management of these identifiers is performed through the use of **Wallets**, which can ensure that credentials and private keys are bound to the intended user or device.

Note that the use of Wallets in the context of SSI has already been proposed for the secure management of information and identities, and actually represents a core building block that is employed for building an identity in the context of an SSI system. Specifically, the **eSSIF-Lab project**, which constitutes an ecosystem of parties that specify, develop, experiment with and validate means in order to support the operation of systems that support the creation of secure identities in order to perform secure online interactions [40, 41], has proposed the use of Wallets in order to store credentials, whether they are self-signed or obtained by issuers of other parties.

In systems that employ SSIs, these Wallets are typically software-based. However, the use of software-based keys creates the challenge of linking an identity with the device it belongs to, as there is no direct link between the key and the device itself. In ASSURED, in order to address this issue, we aim to answer the following questions in regards to the devices participating in the

secure service graph chain: *Is there any way to ensure that each entity is the one that it claims to be? How much trust can we put in statements made by devices or users in regards to their partial identifiers and attributes needed for accessing a service or data?* In other words, *how can someone be sure that a VP really belongs to the correct entity?*

To this end, and in order to shift the responsibility for the assurance of trustworthiness to the devices themselves, we propose the implementation of **Wallets protected through hardware-based keys** in order to safeguard the management of VCs and VPs and enable binding of the Wallet to a specific device. Indeed, with hardware-based support of keys, users are able to create and manage their DIDs using a **Trusted Component (TC)**. That would enable us to transfer the root of trust of the entire SSI ecosystem purely on the digital wallet by considering an underlying Trusted Component as part of the wallet, without making any assumptions on the trustworthiness of the other layers. Our solution could rely on the underlying cryptography of **Direct Anonymous Attestation (DAA)**, which provides platform authentication with strong privacy guarantees. So it can help us achieve not only the vision of a TC-enabled wallet, but also the need of supporting multi-level privacy.

The vision outlined above can be achieved by using any trusted component that can provide the requirements outlined in D1.1 [30] in regards to secure storage and hierarchical key management, and thus has the capabilities to be used as a secure element. However, in ASSURED, the instantiation of the hardware-based Wallet is performed based on the use of a **Trusted Platform Module (TPM)**, since the functionalities provided by the TPM are also used for supporting the various protocols implemented in ASSURED.

As mentioned above, for the participation of a device in the secure service graph chain, the Wallet will have to manage all the necessary **cryptographic materials** required in order to perform the operations required in the context of on-chain interactions or communication between devices. The two main kinds of materials that are employed in these operations are the following:

1. **Cryptographic keys**: A wide range of keys is employed in the hierarchy of each device participating in the service graph chain, such as the **Attestation Key (AK)**, the **DAA key**, or the **secure ephemeral keys** (further analyzed in Chapter 7). These keys will need to be managed by leveraging the capabilities of the hardware-based wallet.

2. **Certificates and credentials**: The aforementioned secure operations performed by devices participating in the ASSURED ecosystem utilize **privacy-preserving attribute-based certificates and credentials**, which should be managed by the device after they are issued by the **Blockchain Certification Authority (CA)**, so that the device can request access to various Blockchain services in a secure and authentic manner.

As it will be analyzed throughout the remainder of this deliverable, in ASSURED we employ the concept of **Verifiable Credentials (VCs)**, that serve to prove the identity of a device (and any other attributes) when communicating with another target device or component or the Blockchain infrastructure. These are defined as a set of credentials that are generated by an **Issuer** entity, who afterwards gives the generated VCs to their intended **Holder**. Next, the Holder stores them and can use them in order to make verifiable claims regarding their identity, by presenting their credentials to a target **Verifier**. In ASSURED, the entity that acts as the Issuer is the **Blockchain CA**, the Holder is the **device itself where the TPM-based Wallet is hosted**, and the Verifier corresponds to **each entity, that aims to verify the data received from a different device, as well as interactions with that device**.

3.1 Creating Trust in Operational or Attestation Transactions

As it was previously mentioned, one of the main challenges that need to be addressed by the ASSURED framework in the context of the communication between an entity and a target Verifier is the verification of the **integrity and the origin of the presented VCs**. Specifically, when a device aims to communicate with the ledger and presents its set of VCs, it needs to be verifiable that the presented VCs really belong to the claimed entity, and not a malicious party who aims to impersonate the original device. If the VCs of the device are purely software-based, it may be hard to link them to the owner device, thus creating trustworthiness issues, not only regarding a specific device, but also the entire service graph chain. Also, there would be no physical binding between the VCs and the device, meaning that there would be no link between the key and a real-world natural or legal person, which would make it hard to prove that the VCs indeed belong to the claimed entity.

The Digital Wallets implemented in ASSURED employ the use of **TPMs** and are able to address these issues, by **integrating hardware-based keys to the Holder's physical device**, thus providing a direct link between the VCs and the legitimate owner's identity. This method provides a mechanism that ensures that credentials and private keys can only be employed by the physical device that the TPM Wallet belongs to, which provides a strong level of assurance on the VC's origin and integrity. As a result, the devices operating within ASSURED are able to prove, in a transparent and automated manner, that their credentials can be trusted, while attesting to their configuration and run-time behavioral correctness.

In ASSURED, the VCs are signed by the DAA key of the TPM hosted in the device, and are verified by the **Blockchain Certification Authority (CA)**. Thus, devices are able to create and manage their VCs using their TPM, which constitutes the underlying **Trusted Component** and acts as a **Root-of-Trust**. By leveraging advanced crypto primitives offered by DAA, we are able to provide the required levels of trustworthiness throughout the lifecycle of digital transactions, thus providing strong assurances and guarantees on the identity of the participating devices during the execution of the attestation tasks performed within the ASSURED environment. Also, this mechanism will make it possible to provide a secure method to **store, share and audit attestation reports** created following the execution of attestation tasks, as well as the secure storage and management of the relevant **measurements and attestation traces**.

3.2 Definitions

The creation and binding of identities to physical devices and users is based on the concept of **Self-sovereign identities (SSI)**. These are defined as digital identities, which do not require a third-party provider to store and centrally manage the data related to the identity of the user or the device in a central database, but the participants in the system operate and self-manage their digital identities in a decentralized manner.

The concept of a **Decentralized Identifier (DID)** is closely related to the conceptual structure of an SSI based system. A DID is essentially a cryptographically verifiable identifier that doesn't require a centralized registration authority, and can be used to identify users and devices. The purpose of DIDs is to shift the control and the responsibility for trust assurance to the devices, by constructing verifiable presentations that can be authenticated by using cryptographic proofs such as digital signatures.

Recall that in the previous section, we briefly mentioned and defined the concepts of the **Issuer**, **Holder** and **Verifier**, which refer to entities that constitute a system that operates based on SSI. Next, we expand upon these definitions, and place them into the ASSURED ecosystem:

- **Issuer:** The entity that issues the **Verifiable Credentials (VCs)**. Note that VCs are essentially certificates that contain verifiable evidence regarding specific attributes of a device. We consider that the **Blockchain Certification Authority (CA)**, which acts as the Issuer in the case of ASSURED, issues the appropriate certificates, so that the device can use them to interact with the Blockchain. The Issuer is the entity who issues the credentials, based on attributes that have been verified by the **Privacy CA**. Note that this process is part of the **device enrollment and registration phase**, which has been presented in D4.2 [34].
- **Holder:** In ASSURED, this refers to the device that hosts the TPM Wallet, which contains the hardware-based cryptographic keys that are required in order to prove the correctness of the VCs and VPs of the device, and are used in order to enable secure interactions between the device and the ledger, as well as between devices. There are various types of keys stored in the TPM. Specifically, the **Attestation Key (AK)** is used in order to sign software or firmware measurements in a **Platform Configuration Register (PCR)**, which are used as part of an attestation process. The **DAA key** is used in order to support user-controlled linkability, which allows a tracer or opener authority to trace back to the DAA signer's ID when required. Finally, **ephemeral keys** provide strong confidentiality guarantees in the data exchange between two devices, or for on-chain interactions.
- **Verifier:** In ASSURED, the term Verifier is used to depict two core roles: The first one is part of the remote attestation process where a Verifier device securely receives the traces from the Prover devices so that it can attest to their correct configuration or execution state [32]. In this context, Verifiers can be any of the deployed devices that can execute all of the attestation enablers offered in ASSURED; e.g., Control-flow Attestation, Configuration Integirty Verification, Jury-based Attestation. The second role has to do with the **Security Context Broker (SCB)** acting as the Verifier of the correctness of the VCs and any presentations and proofs provided by a device (through their Wallet) prior to getting access to a service. More specifically, the SCB is responsible - with the support of the Blockchain CA - to perform the access control prior to granting access to a device to query over the Blockchain services: it first forwards the received VC to the Blockchain CA in order to verify the claimed identity (based on the DAA Credential that was issued and bind to this Wallet and VC during the device registration) and then it checks the valifity of the presneted attributes required for getting access to the target service.

The aforementioned entities can be seen in the conceptual architecture of the TPM Wallet, shown in Figure 3.1. Specifically, the **Blockchain CA**, which acts as the Issuer entity in the ASSURED ecosystem. We consider that the Blockchain CA is **DAA-enabled**, which means that it is able to support the device registration and enrollment process by verifying and activating the device's DAA Credentials as has been issued by the privacy CA [34]. Specifically, the Blockchain CA is enhanced with the capability to provide VCs embodying **evidence on the correct state of the Holder's Device and Wallet**, as checked during registration, based on which the verifiable presentations can be then self-issued. Essentially, the TPM-based Wallet will not allow the Holder device to use any of its VCs if there is any deviation from the expected state. Furthermore, the Blockchain CA verifies the DAA (anonymous) Credential that was issued by the Privacy CA and keeps a record binding this $Cred$ to this wallet so that no other device will be able to present verifiable proofs based on this VC which is binded to this DAA credential. The registration process,

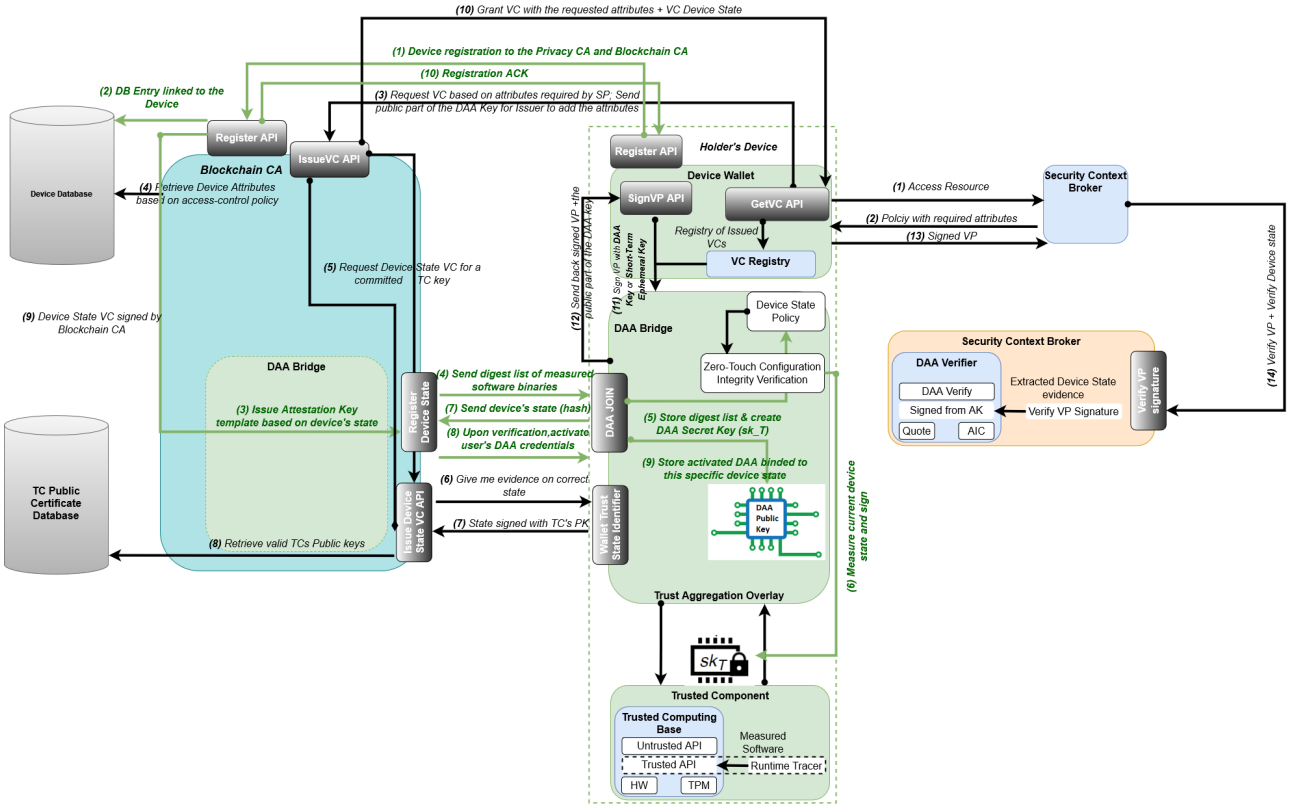

Figure 3.1: ASSURED TPM-based Wallet Conceptual Architecture

as well as the role of the Blockchain CA in the aformentioned processes, will be further outlined and analyzed in Section 3.5.

In addition to the entities mentioned above, another essential component of the TPM Wallet infrastructure shown in Figure 3.1 is the **DAA-Bridge**. This is the component that acts as the middleware that serves to protect the TPM Wallet. It is responsible for protecting the use of VCs, and ensures that the hardware-based DAA key can only be used for signing VPs **if the device's current state matches with the reference value received from the Issuer during registration**. It also provides the advanced cryptographic primitives and attestation enablers, offered by DAA, that can be used for enabling verifiable evidence on the correct state of the Wallet that corresponds to the Holder device, such as measurements on identity and information on the running processes.

3.3 Conceptual Architecture

ASSURED enables the use of crypto operations of the DAA (ECC, Blind Signatures, Zero Knowledge Proofs) towards providing additional trust assurances on the use of VCs: As can be seen in Figure 3.2, issued VCs are binded to a specific Wallet and can only be used if our DAA-Bridge Extension can verify the correctness of the Wallet which, in turn, translates to the respective (private) key of this VC not been compromised. A DAA scheme enables a signer to prove the possession of the issued credential to a verifier by providing a signature, which allows the verifier to authenticate the signer without revealing the credential and signer's identity.

DAA's ECC keys, attestation services and key hierarchy (supported by the underlying TPM and its Endorsement Key that acts as the ID of the Wallet) can support the provision of verifiable

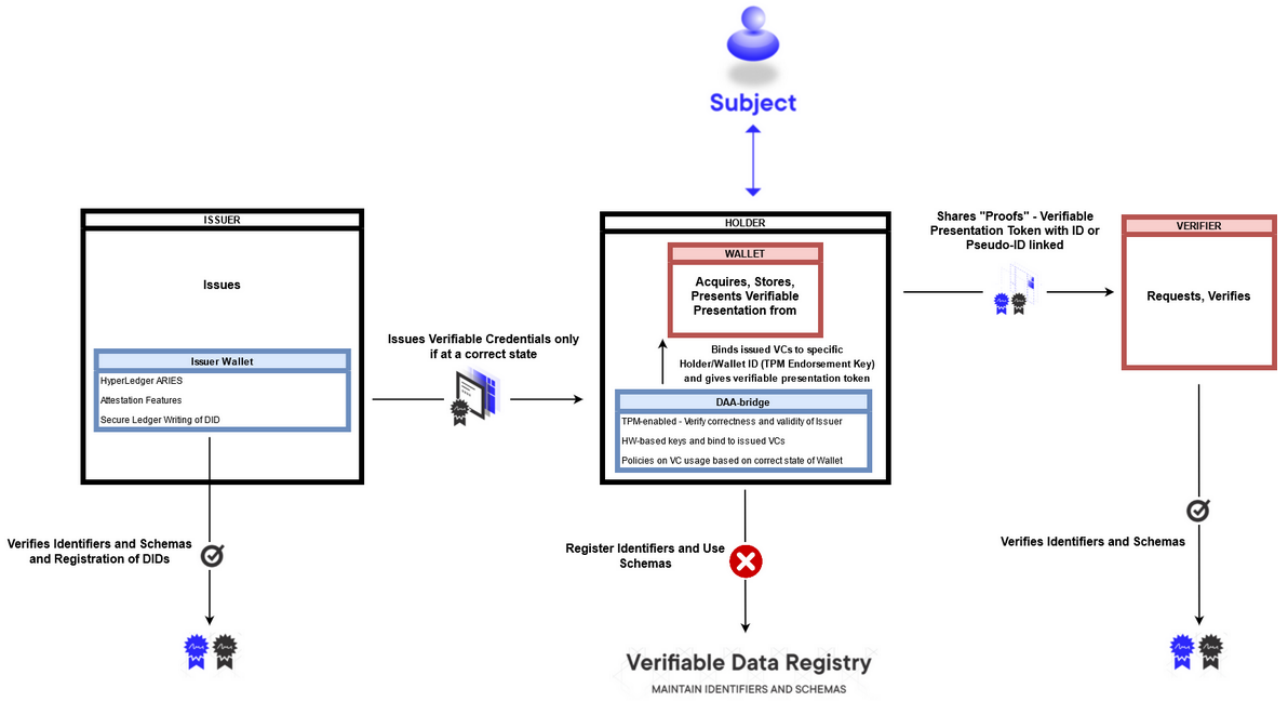

Figure 3.2: ASSURED Identity Management Ecosystem

evidence on the correct configuration state and/or execution of the Wallet. Essentially, the **DAA-Bridge Extension will not allow the use of a key for signing a Verifiable Credential or a Verifiable Presentation unless no compromise has been detected (Device Correct State)**. Only then a signature can be produced based on ECC built on pairing-friendly elliptic curves. Different from a Public-Key Infrastructure (PKI) which is based on a trusted centralized Certificate Authority (CA) to bind public key pairs to identities, DAA certifies identities locally on the Holder.

Overall, we can now enforce that a VC can only be issued by an attested Issuer and that this VC is binded to the Holder's device (wallet), overcoming the current limitations of bare "proof-of-possession" of a sw-based key. ASSURED is agnostic of the wallet's implementation and the underlying VC Data Model considered. Furthermore, ASSURED-enabled Wallets can benefit from the properties offered by DAA when it comes to user-controlled unlinkability: Signer's identity cannot be revealed from a signed VP and multiple signed VPs cannot be linked without the signer's consent. Essentially, the DAA-bridge enables a user to sign VPs (requested while trying to access service resources) under different short-term ephemeral keys that while managed under the same master DAA (private) key, they cannot be linked with each other unless the use configures the DAA-bridge accordingly. This is achieved through DAA's use of both blind and deterministic signatures while signing a VP.

Figure 3.1 illustrates the high-level architecture of the entire ASSURED ecosystem comprising all the aforementioned components, namely **Issuer (Blockchain CA), Holder's Device, and Verifier (SCB)**. The detailed flow of actions that take place in all components for protecting the entire lifecycle of a VC is also depicted highlighting the core features of the ASSURED TPM-based Wallet and its main DAA-bridge module.

Recall that the main objective is to enable the use of hardware-based keys for protecting the issuance and use of Verifiable Credentials (VCs) by a device when trying to access a specific

service resource (Blockchain service or data bundle). The core idea is as follows: DAA-Bridge runs at the Blockchain CA (as the Issuer) and also in the Holder device. These components can directly talk to each other and allow attestations to be digitally signed by the device's hardware-based key (DAA Key) about the device state (its OS and installed apps). *If the device's contents change, the signed hashes will change and the device will need to re-register with the Issuer - otherwise, it will not be allowed to sign any further Verifiable Presentations (VPs).*

In this context, the overall process comprises the following two phases that are highlighted with different colours (are also explained in more detail in Section 3.5):

Registration (*highlighted in green colour*): During registration, the device first of all registers with the Blockchain CA (after having executed the verification process with the Privacy CA so as to check the validity of the TPM) the entry where it can retrieve the attributes (entry in the respective DB) based on which it can issue the requested credentials (Steps 1-2). Then, the DAA-Bridge of the device's aallet is required to attest the device's current state based on a trusted state reference value (Step 3); **digest list containing the hashes of the measured software binaries that are expected to be installed on the Holder's Device.** Upon reception of this digest list (Step 4), and the fact that the DAA-Bridge has created the DAA private key (sK_T as the AK) (Step 5), which is stored in the underlying Trusted Component (TC) so that the host wallet cannot have direct access to it, it then binds it to the received state reference value (Steps 6-9). Dwelving into more details, this process includes the following: After the creation of the DAA secret key (sk_T), the DAA-Bridge of the Holder's Device talks to the DAA module of the Issuer so that it can provide the necessary guarantees of its validity. The Issuer then places a signature on the public key, producing the Attestation Identity Credential (AIC) (public part of the DAA key) creating a certified public part of the Holder's Device AK. This enables the protection of the DAA key (AK) usage - it can only be used for signing a "Device State OK" attribute (Step 6) if the device's current state matches with the reference value received from the Issuer during registration (Step 9) (if something changes in the device's state that has to be recorded with the Issuer again, this can be done through a Re-register Process that will be implemented). The output of this phase, besides the straighforward registration of the User to the Issuer (Step 10), is the creation and binding of the hardware-protected DAA key to the specific Wallet and its usage protection based on a policy (representing the state that the device needs to be for the Wallet to allow the usage of the key) received by the Issuer (Step 9). This enables for the verification of the validity of a "Device State OK" evidence - by the VC Verifier - based on the validity of the attached signature.

Service Resource Request (*highlighted in black colour*): When a device wants to access a specific service at the Blockchain infrastructure or get access to some data already shared, it will be required to present a Verifiable Presentation (VP) accompanied with the appropriate VC (containing the necessary claims/attributes issued by the Blockchain CA) and signed by the registered DAA Key (Step 2). The attributes to be disclosed as part of the VP can be retrieved through the already defined smart contracts [35] that contain the access control policies need to be satisfied so that the SCB can grant permission to a device. This essentially translates to the need of also presenting a proof on the correct state of the Holder's Device to be issued by the Blockchain CA as evidence on the integrity of the Wallet signing the VPs. Upon reception of this policy, the device will contact the Blockchain VA requesting for the issuance of the specific VCs; only in the case that attributes not included in the already provided VCs (issued during the device registration phase) are required. Part of this request will also include the public part of the hardware-based DAA key to which the Blockchain CA will add the requested "claims" (Step 3). Prior to the completion of this task, the Blockchain CA will also initiate the verification process: verify the correct state of the Holder's Device (against the state that was provided during registration) in order to issue the

Device Correct State VC (Steps 5-9). This will create a Quote, signed with the internal DAA Key (AK), that will then be merged with the traditional VC and sent back to the device which will be stored in its VC Registry (Step 10). The Wallet will then sign the VCs into a VP, requesting also from the DAA-Bridge to create an additional signature using either the DAA Key or an ephemeral short-term private key (depending on the privacy requirement) (Steps 11-12), and forwards the VP to the Verifier (Step 13).

Device State Attestation: This operation represents the verification of a "Device State" evidence as part of a VC forwarded by the Holder's Device (as the VC Holder) to the SCB. Essentially, how to prove that a device possesses a VC with a valid attestation attribute when requested. In this case, the DAA Module of the SCB receives TPM Quote (created by the host TPM of the DAA-Bridge of the VC Holder containing the last recorded state of the device), signed by the AK, and the AIC (compiled by the Blockchain CA during Registration Phase - this actually acts as the certified public part of the AK). This package is also signed under the EK of the Blockchain CA. The SCB then invokes the DAA Verify functionality (Section 3.6) for, first of all, checking the correct signing of the TPM Quote (if the Holder's Device is not at a correct state - violating the PolicyDigest - then the Quote will not have been signed by the correct AK) and, if successful, checking that the attached signature is valid on the Quote with regards to the Wallet that holds the correct AK. If the verification of all internal signatures is successfull, then the DAA Verifier replies back OK; FAIL otherwise.

3.4 Functional Specifications

Next, we provide the list of **functional specifications** that must be supported by the TPM Wallet, and we provide a high-level description of the protocols and methodologies implemented in ASSURED, that fulfill each of these specifications. In the left column of Table 3.1, we first present a generic description of the specifications that need to be fulfilled, and in the right column we provide details regarding the instantiation of this specification in ASSURED.

Functional Specification	Instantiation in ASSURED
The Wallet must be able to provide **secure storage** and **secure management** of all cryptographic secret material and keys.	In ASSURED, this refers to the secure storage of cryptographic material required in order to perform secure operations and attestation processes. These may include **Attestation Keys (AKs)** that are used to sign attestation reports, **DAA keys**, **pseudonyms**, and **ephemeral keys**.
Device binding should be supported, while providing guarantees regarding the trustworthiness of the claimed entity. When a device makes a claim regarding its identity, it should be ensured that **the claimed entity belongs to that device**.	In ASSURED, this essentially translates to the ability to **bind a specific Wallet to a device**, meaning that **only the specific device hosting the Wallet should be able to interact with the Wallet**, as well as use the primitives provided by the Wallet. For example, if the Wallet belonging to a particular device extracts a set of VPs, these should not be usable by a Wallet belonging to a different device.
The Wallet should be able to provide decentralized, trustworthy management of **Verifiable Credentials (VCs)** and **Verifiable Presentations (VPs)**.	In ASSURED, the VCs are issued by the Blockchain CA and contain all the attributes relevant to a particular device, while a VP can be extracted by the Wallet based on the VCs, by using the **subset of these attributes** that are required in order to access a specific service. These can afterwards be used in security schemes that require attribute-based authentication, such as **Attribute-Based Access Control (ABAC)** or **Attribute-Based Encryption (ABE)**. The Wallet should be able to manage both the VCs, and the generated VPs.

The Wallet should be able to provide access to the device of such VCs and VPs, **only if it is at an expected, correct state**. In order to respond to a request by a device (for example to create a VP or sign a set of traces), it should be verifiable that the Wallet is in a trustworthy state.	In ASSURED, this is achieved by using **Direct Anonymous Attestation (DAA)**, which is a protocol enabling a TPM and its Host device to authenticate themselves to a Verifier device and to prove that the Host is in a trustworthy state in a **privacy-preserving manner**. Specifically, the DAA scheme provides the TPM with the ability to anonymously sign its **Platform Configuration Register (PCR)** values, while still convincing the Verifier that it possesses valid DAA credentials.
The Wallet must be able to support **enhanced security for allowing secure interaction with the Blockchain**.	The Wallet should be able to create the VPs that are required to perform the **Attribute-based Access Control (ABAC)** scheme, that has been implemented and provided by ASSURED.
The Wallet must support operations that enable access to attestation-related information to stakeholders **with different levels of granularity**, depending on the level of access that is permitted to be granted to each stakeholder.	In ASSURED, the **Attribute-Based Encryption (ABE)** scheme, which leverages the capabilities of the TPM-based Wallet, in order to grant different levels access to a set of attestation data to devices, based on the attributes of the device that attempts to access this information.

Table 3.1: Functional Specifications

3.5 TPM-based Wallet Registration & VC Management Functionalities

Having described the specifications and the high-level architecture of the TPM Wallet and its interactions with the other ASSURED Blockchain components (Section 3.3), we next provide a detailed description of the processes whose operation is supported by the functionalities offered by the TPM Wallet, and serve to achieve the functional specifications outlined in Table 3.1.

3.5.1 Registration, Issuance and Verification Phases

We first start with the **Registration Phase** that is invoked when a device wishes to get (securely) onboarded in the overall network and, thus, it wants to register with the Blockchain CA for getting a representation of the credentials that can be issued (in a verifiable manner) when requested (during the **Issuance Phase**). *All sequence of actions that take place during bootstrap and operation of the ASSURED TPM-based Wallet (Registration, Issuance and Verification), as described in Section 3.3, are depicted in Figure 3.3.*

Note that in the following, the **VC Issuer** refers to the **Blockchain CA**, which is responsible for the issuance of the VCs for each device that either requests registration to the overall network, thus, it authenticates to the Blockchain CA that issues VCs *based on all possible attributes identified for this specific device including also the "Device State OK" attribute, if requested*; or requests the issuance of another VC, during runtime, because of a previously expired credential, a change in the configuration state of the device has occurred (thus, the DAA Key usage policy needs to be updated - Figure 3.4) or additional attributes needed for accessing a service. Conversely, the **DAA Issuer** refers to the **Privacy CA**, which is responsible for the certification of the TPM, the registration of the device and the verification of a device's attributes so that a *verification token* can then be forwarded to the Blockchain for the issuance of the VC. Note that conceptually, in the ASSURED framework, the VC Issuer and DAA Issuer can be a single entity. However, in the

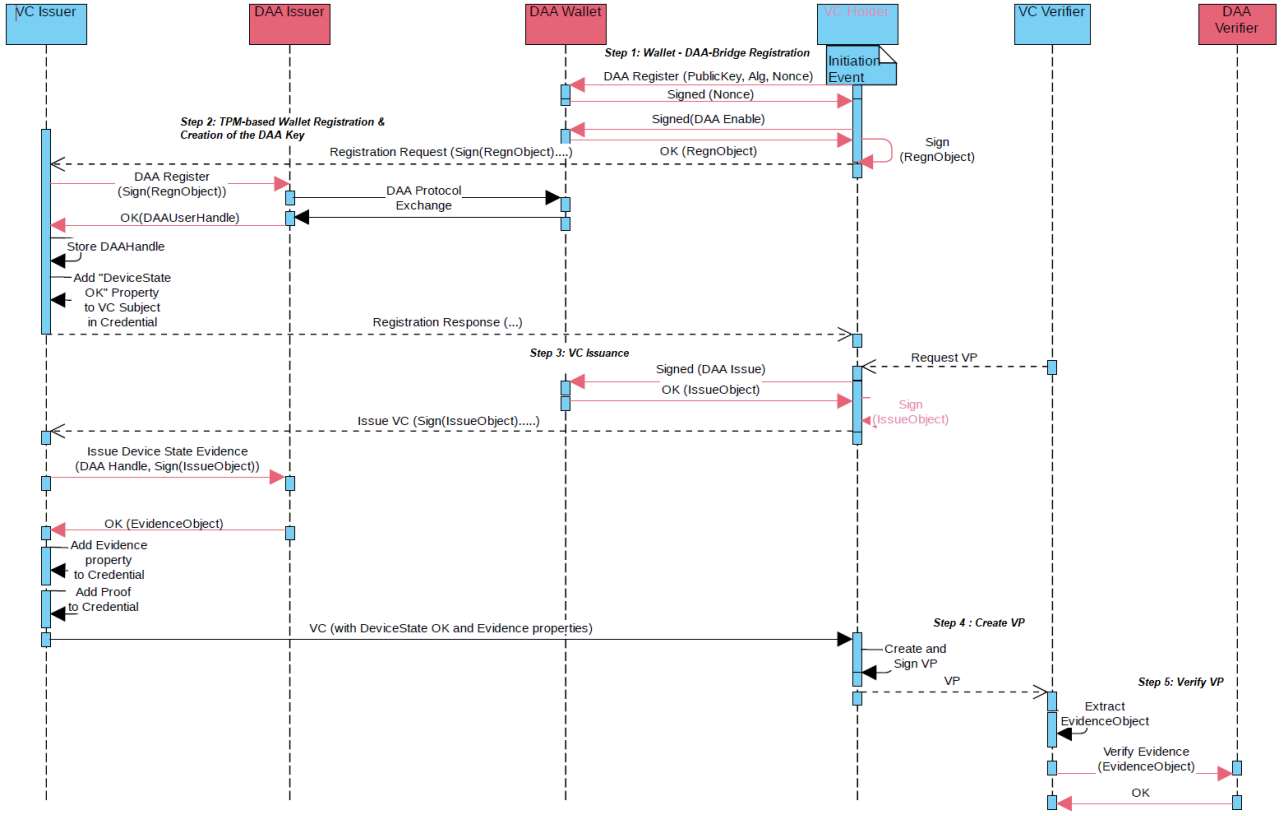

Figure 3.3: ASSURED TPM-based Wallet VC Registration, Issuance and Verification Phases

current design, we have considered them as separate entities that operate in the *least privilege mode* so that no single entity can breach the privacy of the device without colluding each other.

As can be seen in Figure 3.3, consider a Holder device equipped with a TPM-based Wallet that wishes to register to the ASSURED ecosystem by executing the Device Registration and Enrollment Phase [34]. The **first step to be executed is to "*link*" the sw-based Wallet application with the ASSURED DAA-Bridge component** so as to avoid the use of rogue TPMs and/or malicious DAA components that try to intercept the creation of the correct keys for manipulating any future verifiable presentations managed by the device. This is done by registering the Wallet with the DAA-Bridge **(Step 1)** and exchanging their cryptographic material. *We assume that both the Wallet and the DAA-Bridge component come with pre-installed keys that are publicly know for verifying their identity*. So the Wallet will be pre-configured with the DAA-Bridge name identifier while the (genuine) DAA-Bridge component will also be pre-configured with a private key (so all wallet applications can verify its "proof of possession" through the known public key) and the VC Wallet will be pre-configured with the corresponding public key (of a JWK format) and component name. **In this way the VC Wallet will be able to authenticate the DAA Bridge component**.

Then during the registration phase the two componets will do a mutual authentication handshake so that the VC Wallet can authenticate the DAA Bridge app, and vice versa. More specifically, the Wallet creates a **Registration Object** including its public key and a nonce to be signed by the DAA-Bridge component and the TPM. The DAA-Bridge component, upon reception, it signs it using the pre-installed key; note here that this is the key of the component used only for authentication with the Wallet and not the DAA Key that is used for protecting the issuance and management of WPs. When the Wallet verifies the correctness of the received nonce signature then the Wallet and DAA-Bridge component registration has been successful and the DAA-Bridge

will now reply to requests originating only from this specific wallet.

The device, through the Wallet, then proceeds to request its registration to the Blockchain CA by forwarding the **Registration Object (Step 2)**, including also an additional nonce created by the certified TPM, so that appropriate evidence can be given to the Privacy CA (through the Blockchain CA) on the validity of the TPM. The Blockchain CA, receives the registration object, and notifies the Privacy CA to initiate the process for activating and certifying the DAA credential of the specific device's DAA-Bridge component. This essentially includes the **creation and binding of the hardware-protected DAA key** - to this specific Wallet of the device - based on the digest list (trusted reference value comprising the "correct" hashes of the binaries that are loaded in the holder's device, thus, representing the correct, expected state of the device) circulated by the Privacy CA (Steps 3-9 of the Registration flow of actions in Figure 3.2). This allows the Blockchain CA to attest the correct state of the Wallet every time it wants to produce a "Device State OK" credential (Steps 5-8 of the Service Resource Request flow of actions in Figure 3.2). *The exact sequence diagram showcasing the interaction and crypto operations that take place between the Blockchain CA, Privacy CA, DAA-Bridge and TPM are documented in Section 3.5.2 that contains the sequence diagram with all the TPM commands to be executed.*

After the successful creation and binding of the DAA Key and $Cred$, to the devices' wallet, the Blockchain CA notifies the device of the successful finalization of the registration process. This enables the device to then be able to request for the **issuance of the necessary VCs based on all possible attributes identified for this specific device (Step 3)**. This process starts by the device asking to the SCB the list of attributes that are needed for accessing a specific resource; either a Blockchain service or access to recorded (operational or attestation) data. This is achieved through the execution of the $getAttributes()$ function already defined as part of the deployed access control smart contracts [35]. Once the list of attributes has been received, the devices' Wallet first checks to see whether it already has a valid VC (based on which it can create a verifiable presentation disclosing evidence on the possession of the requested attributes) or it needs to request the issuance of a new VC. For the latter, the Wallet creates an **Issuance Request** object signed by the DAA-Bridge and is sent to the Blockchain CA. Then the Blockchain CA requests from the Privacy CA to verify the correctness of the signature which represents that this Wallet has access to the correct DAA credentials. In this case, the Blockchain CA then proceeds with the issuance and transmission of the required VC back to the device. More specifically, invoking the issuance API (Section 3.6) causes the Blockchain and Privacy CA to attest that the Holder's Wallet is still in the correct state (as was registered during the Registration Phase and the binding of the DAA AK), and to return a verifiable credential containing the "Device State OK" subject attribute along with the other device attributes encapsulated in the Evidence property to prove the correctness of all device attributes and partial identifiers. *Again all the detailed operations and actions that take place between the involved entities during the issuance of a VC are depicted in Section 3.5.3*.

We have to note here that possible **re-issuance of a VC might be required either due to the expiration of a previously issued credential or because of a change that occurs in the state of a device.** This might represent a valid change that can be the result of a software update process, for instance, as is one of the core needs in the Smart Aerospace use case towards secure remote maintenance. In this case, when the device needs to create a verifiable proof based on the already issued VC, the **DAA-Bridge will detect the change in the device state and it will not allow the secure issuance and signing of such a VP.** This will trigger the process of the Wallet going back to re-register to the Blockchain and Privacy CA so as to update the DAA Key usage policies, as can be seen in Figure 3.4.

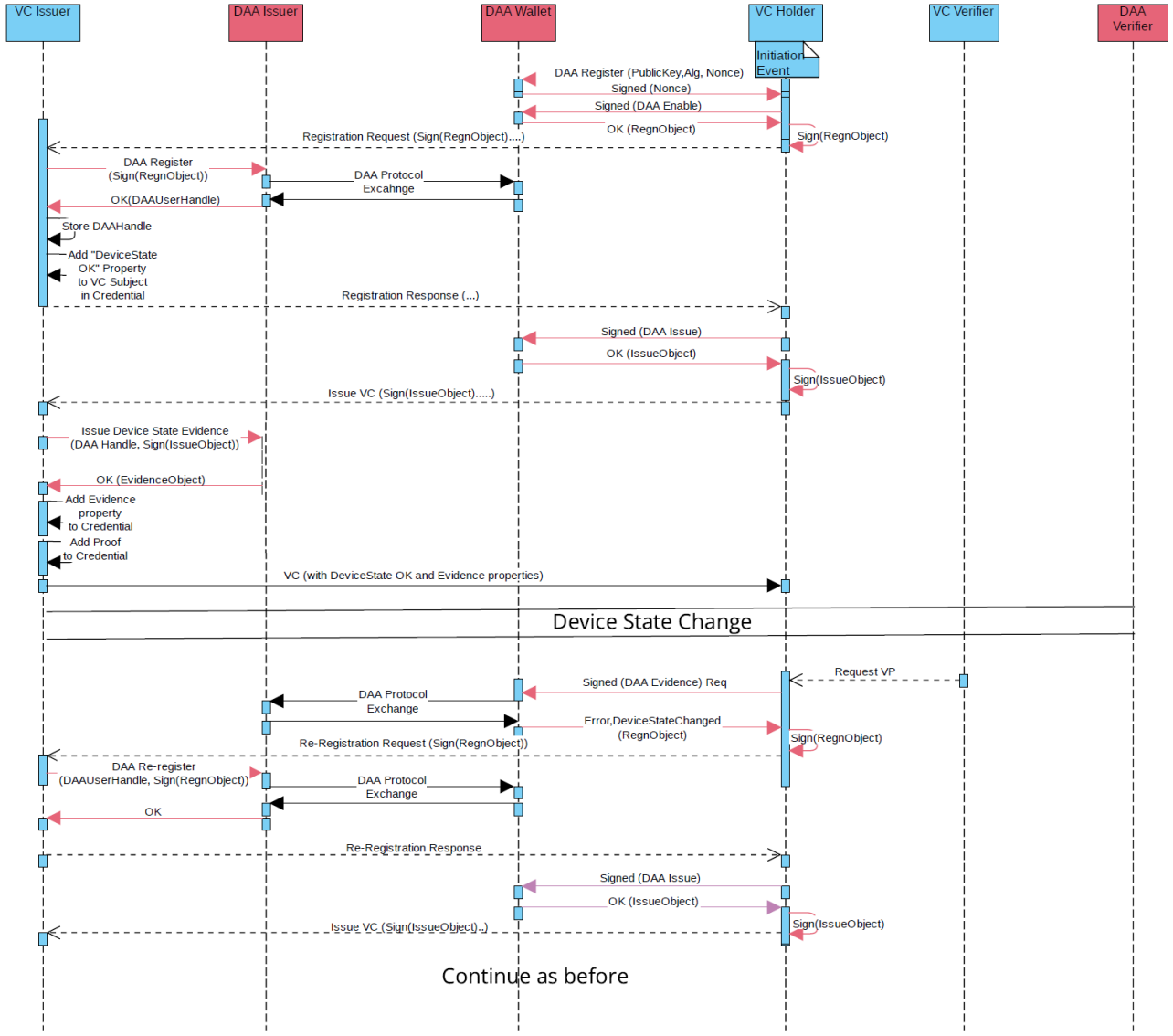

Figure 3.4: VC Issuance when Device State has changed

When a valid VC has been issued to the device, it can then use it to **create the necessary verifiable proofs disclosing only those attributes that are needed for accessing a specific resource (Step 4)**. Prior to creating (and sending) a signed VP, the VC Wallet needs to make sure that this VP can only be used for this specific transaction; *in order to protect against adversaries that may want to inadvertently move (ghost) a VC from this specific holder's device to another device so that they can falsely present evidence attributes for this attacker-controlled device*. This is achieved through the invocation of the DAA SIGN operation for signing a nonce (policyNonce) as provided by the SCB when requested by the device for accessing this specific resource. The provided nonce is signed by the Holder's Device DAA, using the secret DAA Key (AK), as proof of possession in order for the VC Verifier to be able to authenticate the received VP. This signed nonce is then returned to the VC Wallet which is sent to the verifier together with the (signed from the VC Wallet) VP.

Finally, after the successful creation of the VP, the device wallet forwards this to the SCB who, in collaboration with the Blockchain CA, verifies the correctness and validity of the proof including also the validity of all included attributes. This is achieved by invoking the $VerifyEvidence$ API

as described in Section 3.6. More specifically, given a TPM Quote and the AIC (of the holder's Device), the VC Verifier can run a deterministic verification algorithm to, first of all, check the correct signing of the TPM Quote (if the Holder's Device is not at a correct state - violating the PolicyDigest - then the Quote will not have been signed by the correct AK) and, if successful, to check that the attached signature is valid on the Quote with regards to the basename and stems from the Wallet that holds the correct AK.

3.5.2 TPM-based Wallet Registration

In what follows, we give a more detailed description of the crypto operations that take place between the Holder's Device, the Wallet (including the DAA-Bridge), the TPM, the Blockchain and Privacy CA during the **Registration Phase**. In a nutshell, this process refers to the registration of a device to the ASSURED system, where the TPM Wallet needs to register the DAA key that serves to protect the TPM Wallet, which is afterwards bound to a device state that is known to be correct and trustworthy.

In Figure 3.5, we depict a detailed workflow of actions comprising the execution of the **DAA exchange protocol** regarding the creation of the DAA key, as well as the registration of the TPM Wallet with the Privacy CA, focused on the exchange between the Privacy CA and the TPM Wallet.

The action workflow consists of the following steps:

1. The device that wishes to register and certify its TPM with the Privacy CA sends a **Request Registration packet** to its TPM-based Wallet, along with a nonce. The Wallet signs the nonce and sends it back to the Holder device.
2. The Holder device sends the signed registration request to the Blockchain CA, who communicates with the Privacy CA, so that the process of the creation of the DAA key is initiated.
3. The DAA Protocol Exchange between the Privacy CA and the Wallet is initiated. In the beginning of this process, the appropriate function for the creation of the **Attestation Key (AK)** by the Wallet is executed, and a **Policy is calculated over the expected state of the device**.
4. A **Primary DAA key** is created over the aforementioned Policy by the TPM Wallet.
5. After the creation of the DAA key, the Holder device talks to the Privacy CA so that it can provide the necessary **guarantees of its validity**.
6. The Privacy CA places a signature on the public key, producing the **Attestation Identity Credential (AIC)**, which refers to the public part of the DAA key, thus creating a certified public part of the Holder device's AK. This enables the protection of the DAA key usage, so that it can only be used by the Holder device.
7. If the device's current state matches with the reference value used during registration, as outlined previously, **the hardware-protected DAA key is bound to the specific wallet** and a policy that dictates its usage protection is formulated.
8. The Privacy CA **verifies the validity of the signature** and registers the device, by creating a full set of device credentials.
9. The credentials are sent back to the TPM Wallet, and they are stored after being signed by the AK.

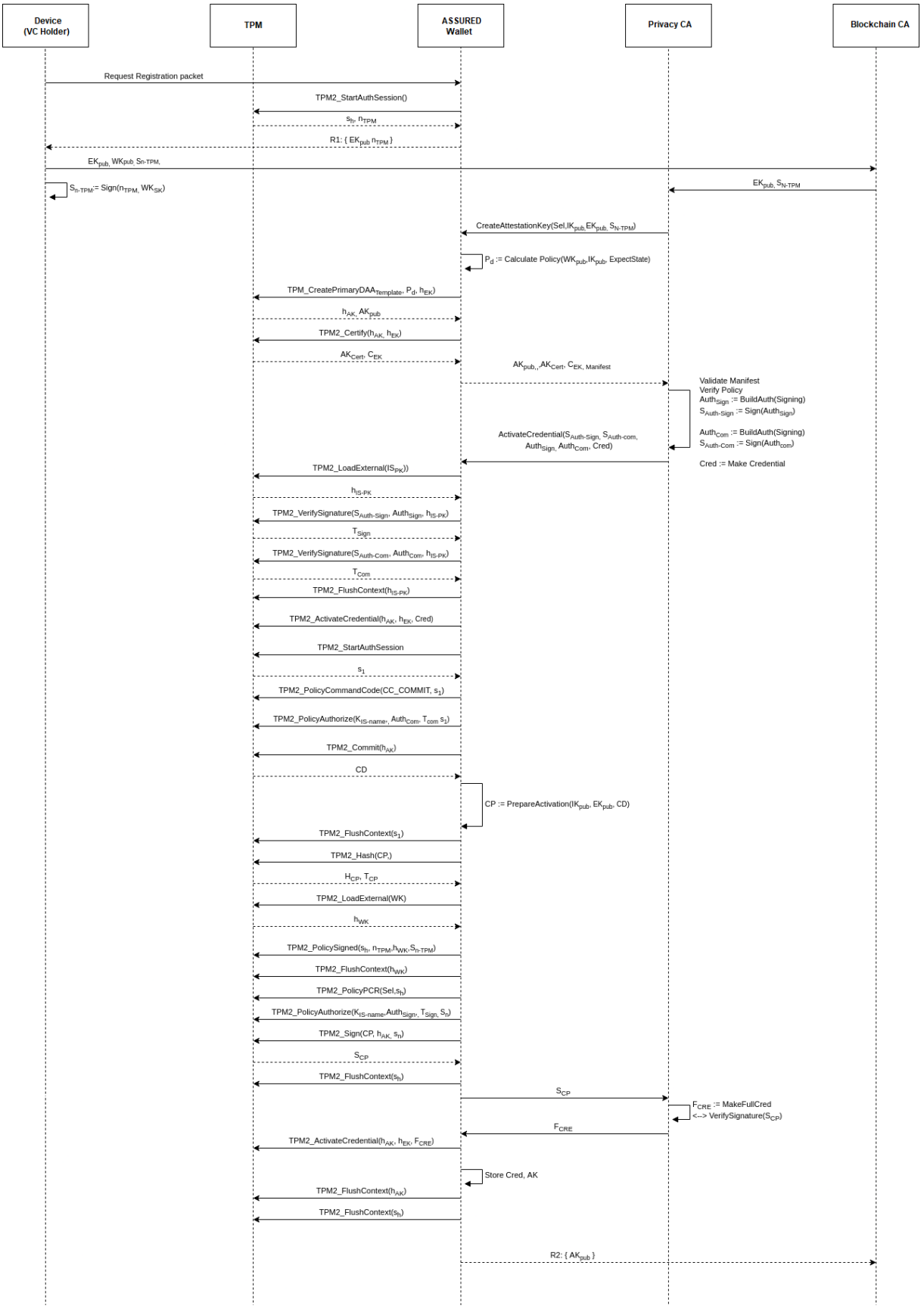

Figure 3.5: Device TPM-based Wallet Registration to Privacy CA

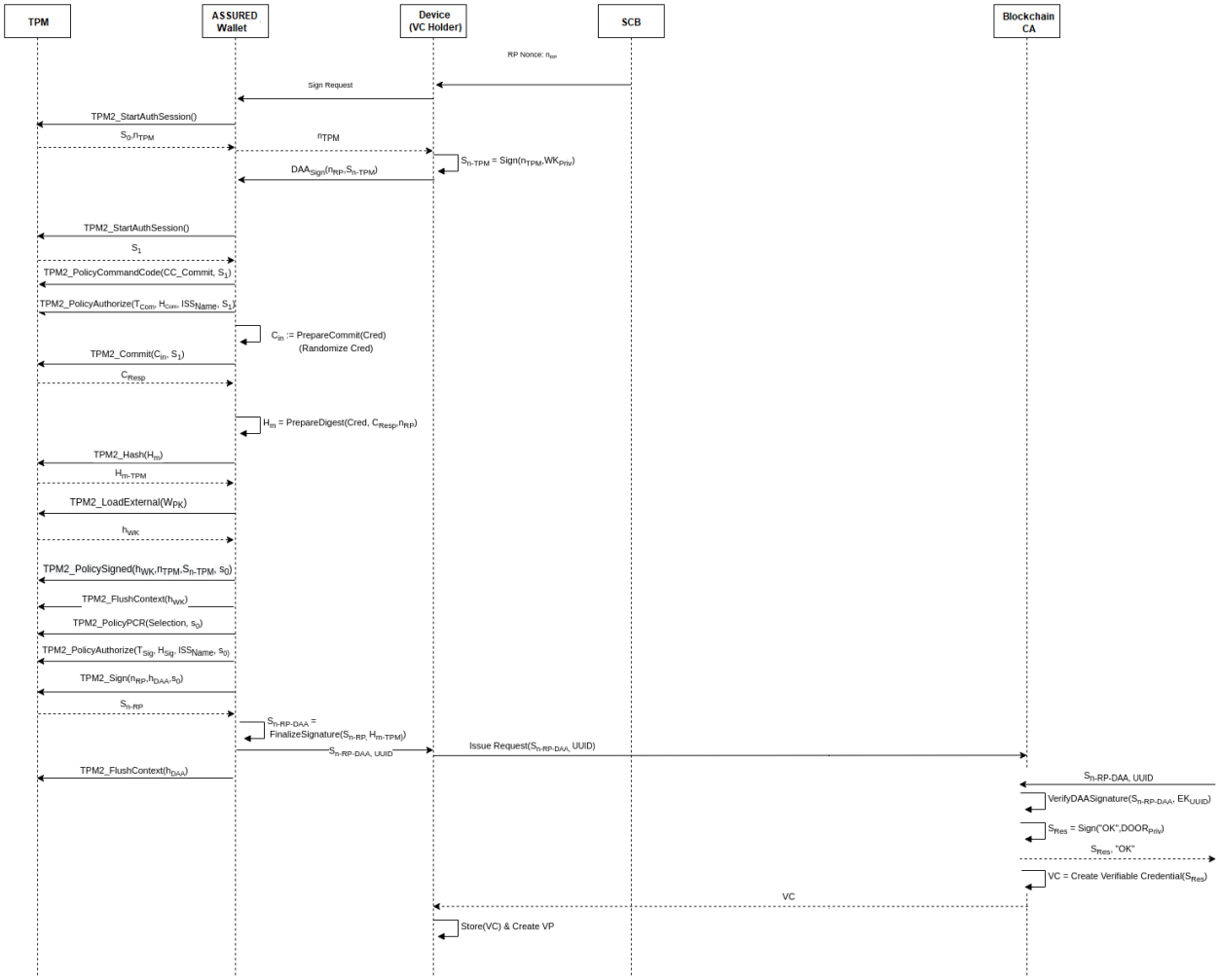

Figure 3.6: Issuance of Verifiable Credentials by the Blockchain CA

3.5.3 Verifiable Credential Issuance & Management

When a device wishes to **access a Blockchain operation or a piece of data on the Blockchain**, as it was previously mentioned, it should be verified that it is in a correct and expected state, based on a list of attributes that is required for this particular service or set of data. Specifically, in order to get access to the desired service or set of data, the device should present the **required list of attributes to the SCB**, so that it can grant the requested access. This is accomplished by using the `getAttributes()` function, which retrieves the list of required attributes from the Blockchain CA in a way that **they can only be used if the device is in a correct state**, so that they can afterwards be **signed by the Wallet using the DAA key**.

First, we present the process followed for the **issuance of the VCs by the Blockchain CA**, which is depicted in Figure 3.6, and consists of the following steps:

1. **The SCB issues a nonce**, which will be used in order to ensure freshness of the transaction and protect against replay attacks, and will forward the nonce to the Holder device.

2. The Holder device issues a **signing request to its TPM Wallet**, and afterwards signs the nonce challenge using the DAA key.

3. Upon reception of the signed nonce, the TPM Wallet **initiates the authentication session**, and then signs a digest on its current status, while also including the nonce received from the SCB in the digest.
4. Based on this digest, **the TPM prepares the request for the desired VCs** and sends it to the Blockchain CA, signed with the DAA key.
5. The Blockchain CA verifies the request. Upon successful verification, it **prepares the requested set of VCs and forwards them to the Holder device**.
6. The Holder device stores the VCs in its TPM. Based on these, it can create the VP required in order to access the service or data that it wants.

Taking into consideration the above process, we can next provide a high-level description of the process followed so that a device can get access to a desired service or set of data. This process can be summarized as follows:

1. When a device wants to access a specific service resource or set of data, it issues an access request to the **SCB**.
2. The SCB requests from the device, in the form of a policy, to present a VP consisting of the required attributes signed with the registered DAA key. This translates to the need to present a VP on the **correct state** and other attributes of the Holder's device as evidence of integrity of the Wallet that will sign the VPs, based on the trustworthy state that has been verified by the Blockchain CA.
3. Upon reception of the policy, the device will issue the required VP disclosing only those attributes needed for accessing this specific resource.
4. In case that the device does not have an issued VC (based on which it can create the verifiable proofs), it contacts the Blockchain CA for getting a credentials based on on all possible attributes identified for this specific device. This is done by invoking the $IssueStateEvidence$ API, as described in Section 3.6.
5. It sends a VC request to the Blockchain CA. The Blockchain CA then engages with the DAA-Bridge of the Wallet in a remote attestation protocol in order to verify the correct state of the holder device prior to issuing the "Device State OK" evidence: It requests from the DAA-Bridge of the Wallet to produce a fresh TPM Quote - holding the device state measurements - and sign it using its AK through the DAA SIGN protocol (the basename used for the random part of the signature can be selected by the DAA-Bridge depending on the privacy requirements for accessing this specific resource).
6. The Blockchain CA will **verify the correctness of the current state of the Holder device**, against the trustworthy reference state that was provided during the registration of the device, in order to issue the requested VCs. The Blockchain CA will then create the required VC and sent back to the Holder device.
7. The TPM Wallet of the Holder device will then **sign the VCs into the required VP**, while also requesting the DAA-Bridge to create an additional signature using either the DAA key, or an ephemeral short-term private key, whose creation process is outlined in Chapter 7. The VP is afterwards forwarded to the SP.
8. The SP **verifies the received VP**, and upon successful verification, grants access to the requested service or data to the Holder device.

3.6 APIs

Finally, this section summarizes the APIs to be implemented for supporting the aforementioned functionalities of the ASSURED TPM-based Wallet for the management of VCs and VPs required towards the continuous authorization and authentication of the devices when interacting with the Blockchain infrastructure. *Details on the implementation of these interfaces will be provided in the second release of this deliverable scheduled for M30 when the final integration version of the TPM Wallet will be available*.

Component-to-Component	Interface	Interface Features
Wallet - (Wallet) DAA Bridge	**DAAEnable API**	Prior to the Wallet sending a **registration request** to the Blockchain CA, it calls the DAA Enable API to check the status of the internal DAA Bridge, and to request the **registration package** comprising the commit value (based on the AK), the Public Part of the AK coupled with the Public Part of the Endorsement Key (EK), of the host TPM that generated the AK, as well as the unique name of the DAA Bridge. All these attributes are signed by the AK for further verification by the Blockchain CA.
Blockchain CA - Privacy CA	**DAARegister API**	The Blockchain CA passes the **registration package** to the Privacy CA for initiating the DAA JOIN phase with the DAA-Bridge of the Holder's Device. All messages between the Privacy CA and the DAA-Bridge will be exchanged through a relay server. The output of this JOIN phase will be stored in the TC Database in a new entry created for this specific device. However, if the device has already registered, the entry will already exist in the database, so the entry handle will already be known. Re-registration is required when the expected correct device state has changed, which may be triggered by the installation of a new application binary. If the DAA JOIN phase is performed successfully, the Privacy CA records **the AIC**, which is sent back to the DAA-Bridge of the Holder device, as well as **the Quote** and **the public part of the AK**, in the TC storage. The Blockchain CA will be notified if this registration was performed correctly.

Table 3.2: Registration Phase APIs

Component-to-Component	Interface	Interface Features
Wallet - (Wallet) DAA Bridge	**DAAIssue API**	When the Wallet wants to request VCs from the Blockchain CA, it will invoke the DAA Issue API to get back **the name of the DAA-bridge**, signed under the AK, **the AIC** that was compiled by the Privacy CA during Registration for this AK, plus **a nonce to guarantee freshness of the request**. This is called the **issue package** and it will be returned as a JSON object to the Wallet for it then to be forwarded to the Blockchain CA.
Blockchain CA - Privacy CA	**IssueState Evidence API**	When a device wants to request the issuance of a VC from the Blockchain CA, it sends an issue request, containing **the list of attributes it wants**. The Blockchain CA invokes the **Issue State Evidence API**, adds a nonce, and provides the **issue package** to the Privacy CA, plus the **user handle**, extracted from the VC database. The Privacy CA retrieves the AIC from the TC Database, checks it against the received AIC, and verifies the validity of the nonce.

		Then, it performs a **remote attestation protocol** with the DAA-Bridge of the Wallet, in order to verify the correct state of the Holder device, prior to issuing "Device State OK" evidence. Specifically, it requests from the DAA-Bridge of the Wallet to produce a fresh **TPM Quote** containing **device state traces**, which is signed using the AK through the DAA SIGN protocol. The Quote also contains a nonce sent by the Privacy CA. If the Quote is valid, then the Privacy CA will issue **VC State Evidence** for the VC linked to the specific AIC.

Table 3.3: Issuance Phase APIs

Component-to-Component	Interface	Interface Features
Wallet - (Wallet) DAA Bridge	**Sign API**	Prior to creating (and sending) a signed VP, the Wallet needs to make sure that **this VP can only be used for this specific transaction**. in order to prevent potentially malicious actors from impersonating the device, or sending falsely present evidence attributes for the attacker-controlled device. This is achieved by invoking the DAA SIGN operation for signing a nonce, as provided by the Service Provider. The **nonce is signed by the Holder's Device DAA**, using the secret **DAA Key (AK)**, as proof of possession, so that the VP can be verified. This signed nonce is then returned to the Wallet, which is sent to the Verifier together with the VP, which has been signed by the Wallet.
Wallet - SCB	**VerifyEvidence API**	The SCB receives the VP from the Holder device disclosing only those attributes required for accessing a specific resource. Then, the SCB forwards the VP to the Blockchain CA for verifying the *correctness of the signature* based on the attached DAA credential, and if successful it notifies the SCB. The SCB extracts the **Evidence Properties** from the embedded VP. It checks the validity of both the **Device State blob** (included in the evidence) but also that this VP was intended for the specific AIC, by executing the DAA VERIFY process. This blob contains the **TPM Quote** created by the Holder device signed by the AK, and the **AIC**. This package is also signed under the EK of the Blockchain CA. If the verification of all internal signatures is successful, then the SCB proceeds in checking whether all the required attributes (based on the access control policy) were disclosed as part of the VP.

Table 3.4: Verification Phase APIs

Chapter 4

Interconnection of ASSURED Blockchain Components to the TPM-based Wallet

After having defined the detailed architecture of the ASSURED TPM-based Wallet, in what follows, we proceed with a description of the security functionalities and services that rely on the integration of the Wallet and its interaction with the other components in the overall ASSURED ecosystem. Recall, that the main functionalities for which the Wallet acts as the *trust anchor* are the following: (i) **Creation, management and protected usage of all necessary cryptographic material (keys)**, (ii) **Management of the VCs** (issued by the Blockchain CA during the device registration and enrolment phase) and **VPs required for the continuous authentication and authorization (Attribute-based Access Control)** of the device when trying to access a resource (Blockchain service and/or data recorded on the ledger), and (iii) Provision of the required **crypto operations for enabling the creation and of the necessary ABE keys**, binded to specific device attributes, for encrypting the raw system data (traces) prior to been uploaded to the ASSURED Data Storage Engine [31, 34]. Therefore, it is obvious that all of these features rely on the presence of the underlying Trusted Component.

The concept of the **Trusted Platform Module (TPM)**, which was defined in [3], is one of the core building blocks for digital security solutions. A TPM provides a fundamental set of low-level capabilities that can be leveraged by higher layers of the cybersecurity chain. Specifically, the TPM can be used in order to **store keys**, **ensure authentication**, **support attestations**, and perform many other tasks in a secure and privacy-preserving manner. It also supports secure and flexible key management solutions, with the implementation of a **lightweight multiple layer key hierarchy** with a very small amount of internal memory. With the key hierarchy architecture, the TPM can securely store cryptographic key material, which provides the basis for building IoT security solutions. The nature of hardware-based cryptography ensures that the information stored in the TPM is better protected than software-preserved data.

The TPM also serves as a **Root-of-Trust**, i.e., a trust anchor for the host platform where it is embedded. It can be used as the basis for the application of attestation protocols, which provide evidence regarding the trustworthiness and the correctness of the operational state of the host system, e.g., by certifying the boot sequence the host is running on. These attestations are required by the ASSURED use cases in order to enable remote security attestation, and to enhance the confidentiality and integrity of the exchanged data.

Nowadays, TPMs are present in many laptops and desktop personal computers (PCs), and are also used by enterprises for tasks such as secure disk encryption. However, they have yet to be incorporated to any significant extent in smartphones, game consoles, televisions, in-car computer systems, and other computerized devices and industrial control systems. To this end, it is

possible to envision and implement systems that utilize the capabilities of TPMs, in order to provide security and privacy assurances for such devices belonging to large and complex Systems-of-Systems, as well as the trustworthiness of the entire interconnected device ecosystem.

In this direction, ASSURED will significantly advance the **state-of-the-art of Blockchain attestation and verification methods** that can achieve these goals. While most existing methodologies rely on computationally costly proofs-of-work, ASSURED will instead use TPMs as the central building block, in order to build a resource- and computationally-efficient **two-stage Blockchain verification mechanism** for achieving the desired trustworthiness level of a system, which will be suitable even for resource-constrained devices, such as mobile devices equipped with a TPM. Specifically, the two stages of the envisioned methodology are as follows:

- In **Stage 1**, the device will perform a Blockchain update, and will then compute the hash value of the updated Blockchain state, based on the ledger updates and the hash of the current state.
- In **Stage 2**, the chosen Verifiers will be able to verify the trustworthiness of this update. If the verification is successful, the users will then replace the hash of their current state stored inside their TPMs with the updated one.

The ASSURED design will also include the implementation of the appropriate cryptographic trust anchors in the hardware domains with the use of TPM technology, as well as software-based Roots-of-Trust and efficient cryptographic primitives [34], for the secure operation of digital trust across a community of trust in a supply chain ecosystem.

In this chapter, we describe the main functionalities of the TPM wallet within ASSURED Blockchain services. TPMs constitute the basis for trusted Blockchain wallets that will be adopted in the ASSURED framework to (i) provide strong **user authentication** and securely store the device credentials based on the TPM's secure key storage functionality, (ii) **control and authorize access to private or public ledger channels** based on the authentication process (e.g., to authorize access to or operations on different ledgers), (iii) securely and efficiently **verify Blockchain updates**, (iv) continuously **attest to the security and trustworthiness of all involved devices** in a privacy-preserving manner, and (v) be able to handle basic **ledger operation fees and digital transactions**. We will expand on the implementation of the wallet in the next version of this deliverable [36].

4.1 ASSURED Services Enabled by the TPM-based Wallet

4.1.1 TPM Wallet Attestation Support

As previously mentioned, the TPM trusted hardware technology is required by the ASSURED use cases in order to enable remote security attestation, and to enhance the confidentiality and integrity of the exchanged data. The trusted Blockchain wallet architecture allows the safe deployment and verification of software updates (as needed in the context of the "*Smart Aerospace*" use case), which often are a source of malware and other attacks.

Blockchain wallets can also be used in order to make statements regarding the trustworthiness and correctness of the state of the devices, as well as the overall system. For example, a trusted platform can attest to a measurement to show that a particular software/firmware state exists in

a device. **This attestation takes the form of a signature over a software/firmware measurement in a PCR using an Attestation Key (AK) protected by the Blockchain wallet**. An external entity may also attest to a platform, in order to confirm that the platform contains a **Root-of-Trust-for-Measurement**, plus a trusted path between the RTM and the Blockchain TPM-based wallet. In the ASSURED framework, devices are required to perform attestation tasks through their trusted TPM Wallets, in order to provide **assurance claims** to show that they maintain specific properties that may be required for accessing sensitive data or other services.

In ASSURED, we consider the **Trusted Network Connect (TNC)** open architecture for network access control [57], based on which we can leverage the functionalities provided by the TPM in order to define a strategy to control the process with which a device can obtain access to the system. Specifically, if a device wishes to gain access to the network, it should be able to prove to another edge device that it possesses a genuine trusted platform, and is in a trustworthy state. Consider, for instance, the "*Smart Manufacturing*" use case, where the IoT Gateway - as the data aggregator of the location data produced by the deployed sensors as part of the Real-Time Monitoring System (RTM) - needs to be able to provide strong guarantees on the correct configuration state of device prior to securely on-board it to the overall manufacturing floor. This can be achieved through the use of the Trusted Wallet, which provides the basis for the execution of the relevant **attestation policies** in the form of **smart contracts**. The integrity of the execution of a smart contract in the ASSURED framework can be verified by producing hash chains required by the Blockchain infrastructure, so that an input can be authenticated by using previous hash key lists.

To protect the security and safety profile of the entire ASSURED ecosystem, system components must be able to make and prove statements about their states and actions, so that other Cyber-Physical Systems-of-Systems (CPSoS) can align their actions appropriately and an overall system state can be assessed and security policies can be evaluated and enforced. In the ASSURED Blockchain architecture defined in [29], the aim of these attestation schemes is to prove to any remote party that an edge device, **its configuration and running services are intact and trustworthy**. The latter is achieved through the **secure recording, sharing and auditing of all attestation reports** leveraging ASSURED's policy-compliant Blockchain infrastructure. Specifically, the ASSURED framework asks for an attestation report that allows the Blockchain node to verify that a device has a legitimate TPM, that reports a correct attestation record.

For instance, upon executing a smart contract to create a **Control Flow Attestation (CFA)** about the state of an edge device, a TPM wallet receives an attestation challenge, then responds to the challenge by making traces of the state of the edge device, and updating the relevant PCR values with a signature on these PCRs. Any device which acts as a verifier then checks if the reported PCR value matches with one of the accepted PCR values "White-List", and verifies the TPM's signature. This kind of attestation is called property-based attestation. The goal of such attestations is to convince any Blockchain device, which acts as a verifier, that the edge device it is communicating with is running on a trusted TPM wallet and using the correct software. After authenticating and attesting edge devices by with the help of verifier devices, the **Security Contet Broker (SCB)** places the report of the execution, which is usually a signature done on the PCR, on the ledger, so that it will be accessible by other edge devices and/or stakeholders.

In D3.2 [32], many efficient attestation schemes were presented and their security was analyzed. These schemes aim to convert the Blockchain devices to trust anchors, capable of providing verifiable evidence for their correct configuration and execution covering both (privacy-preserving, if needed) device authentication purposes (secure enrollment of a device that wishes to join a CPSoS), as well as the operation of the device throughout the lifecycle of the system. This can

include the trusted boot and integrity measurement of a device, enabling the generation of static and run-time behavioral attestations of safety-critical components of a system, and providing strong guarantees on the correctness of the control flow properties. For this purpose, ASSURED provides a **layered attestation toolkit**, leveraging purely SW-based tracing and introspection capabilities, enhanced remote attestation mechanisms and advanced cryptographic primitives for guaranteeing the required privacy requirements. We further expand on these mechanisms in D3.2 [32].

In many cases, it is more efficient to attest multiple devices simultaneously rather than each device separately. This is achieved by the Swarm attestation methodology presented in D3.6 [33]. In this method, multiple devices communicate with the Blockchain Peer, whose functionality has been described in detail in [35], and the Peer aggregates the attestations, in order to attest to the trustworthiness of the swarm. These attestations can either be achieved through ordinary signatures, or through complex anonymous signatures, by employing a mechanism known as **Direct Anonymous Attestation (DAA)**. This is required for preserving privacy in some attestations [12, 13], which may involve attesting a potentially large number of devices as in Swarm Attestation. A main design goal of DAA is that attestations are made in a **privacy-preserving manner**, meaning that any verifier can check that attestations originate from a certified hardware token, but no information regarding the identity of the TPM is divulged. Another important feature of DAA is that it supports user-controlled linkability, which is driven by a **base-name (bsn)**. In the ASSURED Swarm attestation, we aim for DAA with an additional security property referred to as **traceability**, which allows an authority called a tracer or an opener to trace back the DAA signer's id when required.

Swarm Attestation starts at the level of the IoT devices, which perform the attestation and report the result to edge devices. The edge devices aggregate the attestation results they receive from the corresponding IoT devices, accumulate with their own attestation results, and the final attestation results are reported to the central ASSURED components. In the Swarm Attestation scheme, the attestation result enables a centralized Verifier to validate the overall state of the network in a privacy preserving manner, but with the property of tracing back any corrupted edge device or IoT device relying on the ASSURED DAA with a traceability protocol. Such an attestation outputs a Boolean result (i.e., 1 if all devices are healthy and 0 otherwise), without precisely identifying compromised devices. This Swarm Attestation scheme aims to provide privacy-preserving guarantees for both IoT and edge devices.

As previously mentioned, ASSURED provides techniques for achieving the secure enrollment of devices, which has been explained in detail in D4.2 [34], as well as ensure the correctness of their run-time operation. However, it also provides services that act as a second line of defense, when inconsistencies are identified between the results reports by proving and verifying devices and further investigation is needed for detecting whether a device has performed a faulty attestation. This is achieved through **Jury based attestation**.

In regards to the run-time attestation of the devices, ASSURED employs a methodology referred to as **Control Flow Attestation (CFA)**. This involves the verification of traces extracted during the execution of various processes on the device, and will be implemented by leveraging the capabilities of the TPM-based wallet, working in tandem with the **ASSURED Tracer**. The Tracer is a trusted component that measures the current state of the device's configuration, which may include its base software image, platform-specific information, and other binaries.

4.1.2 Secure Download & Execution of Policies/Functions towards the Execution of (Attestation) Smart Contracts

Smart contracts in ASSURED are deployed to the ASSURED DLT network via the **Security Context Broker (SCB)**. These smart contracts are the instantiation of policies and attestation functions which are expected to be applied locally by the different edge devices, in order to perform various attestation tasks. To achieve this, an edge device has to request access to the ledger's contents, and once the DLT network provides this access to the edge device, the latter is able to read and download the different policies and functions in order to start the execution of the respective attestation process.

The workflow of actions starts with an edge (Verifier) device that is about to perform an attestation function (triggered by a smart contract) and then needs to provide the results back to that smart contract, as explained in [27]. The Verifier device makes use of its embedded trusted TPM Wallet, which is the one that will eventually retrieve the calculated policies and will initiate the remote attestation process for the target Prover (all structures defined for the Attestation Policy, Attestation Task and Attestation Report depicted through smart contracts can be found in [35]). In this line of events, the TPM wallet first executes the necessary function to get the attributes discussed above, triggering the `GetAttributes()` method, in order to see the list of attributes that needs to disclose through the constructed VP (Section 3.5.3).

Once this function is executed, the TPM Wallet will then create the appropriate **Verifiable Presentation** based on this extracted list of attributes. Specifically, via this method, a request is sent to the Security Context Broker to access the designated resource (attestation policy or attestation report). The SCB by invoking the `GetAttributes()` method, retrieves the list of attributes required to be exhibited by the requesting actor and sends this list back to the device's Wallet. The Wallet, as described in Section 3.5.1, it first checks to see if it has a valid VC based on which it can create the necessary VP, signed by the Wallet's DAA Key. If not, it first interacts with the Blockchain CA for getting a VC based on all the attributes of the device that have been verified by the Privacy CA. Otherwise, it creates the VP which is forwarded to the SCB for verification. The SCB, upon reception, it first forwards this VP to the Blockchain CA for verifying the DAA signature based on the attached DAA credential so as to make sure that this is the correct TPM Wallet that issued the VP. If this check is successful, then the SCB checks the correctness of the included attributes prior to granting access to the device to download the policy and/or report.

4.2 Secure On/Off Chain Interactions Supported by the TPM-based Wallet

Next, we present the protocol that leverages all the cryptographic operations provided by the TPM-based Wallet to enable the aforementioned services, such as the **Creation of Attestation Reports**, the **Secure Recording of Attestation Results on the Ledger**, and the **Authenticated and Authorized Querying of Attestation Data (from the Ledger)**.

The workflow of actions followed during an attestation process, focused on the perspective of the functionalities provided by the TPM-based wallet, is as follows:

1. The first step of the workflow is the **Authentication of the TPM-based wallet by the Blockchain Peer**. This is be done through a **Message Authentication Code (MAC)**, which

is a form of symmetric signature over some data, in addition to Attribute-based Access Control. Authentication provides assurances that protected data was not modified, and that it originates from an entity with access to a key value that needs to be a secret or a shared secret. During communication between the TPM wallet (from the verifier device) and the peer, the peer authenticates the device that acts as the verifier. In the context of ASSURED, the verifier should be authenticated before requesting an attestation report from a prover device.

2. After authentication of the TPM wallet by the Blockchain peer is performed, the verifier device requests attestations about the state of the prover device. For example, a trusted platform can attest to a set of measurements, in order to show that a particular software/firmware state exists in a device and it is running as expected. This attestation takes the form of a conventional digital **signature** over a software/firmware measurements using an attestation-key AK protected by the TPM-based Wallet. The output of this procedure is an Attestation Report AR.

3. Any TPM wallet can ascertain its correctness in an **oblivious** manner by using a simple challenge-based remote attestation protocol, referred to as **Attestation by Proof**. To initiate an attestation, the verifer uses its TPM-based wallet to sign a challenge ch that is extracted by the attestation policy from the Blockchain. The aim of this attestation is to prevent static code injection, malicious software updates and malware. This also includes the attestation of dynamically loaded libraries and hardware components. To attest the dynamic properties, the data and control flow traces provided by the ASSURED Tracer need to be attested. The verifier then sends the signed attestation challenge to the prover device, where the **Attestation Agent** verifies the correctness of the signature on the challenge under the public key of the verifier.

4. If the verification of the verifier's signature on the challenge is successful, then the prover initiates the attestation through its TPM wallet that obtains a trace-set $\mathcal{N}$ from an authorized tracer, which is to be signed by the TPM wallet Attestation Key (AK) that has a restricted attribute which states that this AK can only be used to sign traces that originate from an authorized tracer. The tracer initiates an **authorization session** with the TPM wallet. If authorization is successful, then the TPM-based wallet will proceed by signing the traces $\mathcal{N}$ using its AK and creates its Attestation Report AR.

5. The prover then creates DAA signature on the Attestation Report AR. Note that the TPM attestation DAA key and its associated credential are not sent to the peer. Instead, the TPM generates a **Zero Knowledge Proof (ZKP)**, that shows that the TPM generates an attestation using a legitimate key DAA with a valid credential that is linked to the AK used to sign the traces, while hiding the identity of the TPM-based wallet.

6. The TPM-based wallet verifies its attestation results and hashes the attestation report AR using the ASSURED defined **collision resistant hash function** H, that outputs $H(AR)$.

7. The TPM-based wallet provides extra information about the attestation log files. The TPM-based wallet encrypts, using the our designed **Attribute based Encryption** ABE protocol based on [63], the attestation log files (metadata) MD under its attribute keys Key_{att} provided by the SCB in the registration phase.

8. In case of successful attestation, the TPM-based wallet sends a **"Successful Attestation"** string to the peer along with $H(AR)$, as well as other information such as attestation log

files, the Device ID, a serial number for each attestation process, the time stamp on the attestation execution, etc.

9. The TPM-based wallet sends the encrypted metadata $E(MD, Key_{att})$ along with the hashed attestation results $H(AR)$ to the the **API Layer**, which resides within a Peer in the private channel.

10. The Blockchain peer verifies the authenticity of the TPM wallet input $\mathcal{N}$, in order to ascertain that it originates from the tracer. This is done by verifying the TPM's signature under the restricted TPM AK. In case of DAA, the peer verifies the DAA signature, i.e., it verifies the ZKP generated by the TPM-based wallet. Finally, the peer outputs its verification results based on the correctness of the attestation result AR. Then it forwards $H(AR)$ together with $E(MD, Key_{att})$ to the Security Context Broker, which stores $H(AR)$ on the private ledger and the encrypted ciphertext in the **Off-Chain Storage facility**, and gets back a pointer to the exact storage location of the data. The pointer is encrypted under the SCB encryption key Key_{SCB} and transformed into a searchable index to be used by the **Searchable Encryption** component.

11. The TPM-based wallet will be provided, by the Blockchain Peer, with a transaction that contains the block address of the hashed attestation results on the private ledger.

12. When a device or user wants to (i) **access data on the ledger**, such as an attestation result, or (ii) needs to get access to any piece of information that is stored **off-chain**, such as attestation evidence whose location is indicated by a pointer stored on the ledger regarding this particular attestation, the authentication of the requesting device needs to be performed through the TPM wallet, by performing **Attribute-Based Access Control (ABAC)**, which is executed on the SCB. This is a process that verifies whether the requesting device has the appropriate attributes, that are required in order to access this specific piece of data. As part of the ABAC process, the TPM wallet needs to execute the following functions so that the device can obtain access to the desired information:

    - The TPM-based wallet interacts with the Blockchain Peer in order to run the GetAttributes() function. This function returns the list of attributes that the device should exhibit in order to obtain access to this specific result. Based on these attributes, the TPM wallet constructs a **verifiable presentation**. Recall that, during registration of a device, a set of **verifiable credentials** is generated by the Certification Authority, and contains a set of attributes of that device. The verifiable presentation constructed by the TPM wallet should contain the subset of the attributes, needed in order to perform ABAC in this particular case. Note that this verifiable presentation should be constructed in a secure manner and signed with the DAA key.
    - The verifiable presentation is then sent to the SCB. Afterwards, through the verifiable presentation, the SCB checks whether it has been correctly signed with the DAA key, in order to ensure that it originates from the correct device. Then, the SCB sends the verifiable presentation to the **Membership Service Provider (MSP)** so that it can check that the attributes in the received verifiable presentation are correct, based on the attributes issued by the Blockchain CA. Recall that the **Blockchain CA** has previously verified the DAA signature and issued the device attributes. If this check is successful, then the MSP notifies the SCB that it can proceed with giving the device/user access to the requested data. The SCB will also determine the required access level (e.g., read, decipher, write, decrypt), send a pointer to the TPM based-wallet to indicate the

location of the encrypted raw data and/or the attestation report in the public and/or private ledger, respectively, and instruct the relevant modules of the system to execute the request.

13. Based on the response received by the SCB from the Blockchain CA when performing the attribute check as part of the ABAC process, it is determined whether the device can proceed with the query for the required information. Then, through the Blockchain Peer, a query function can be executed on the smart contract, in order to obtain the desired attestation result. From the above, it is evident that it is impossible for a device to bypass the ABAC process in order to obtain access to the private ledger. Specifically, the device can leverage its TPM wallet to obtain information on the **private ledger** as follows:

    - Once a TPM-based wallet is allowed to access the network, a **search query** is sent to the Blockchain peer that handles the requests of the channel. The peer, working in tandem with the SCB, can resolve whether the TPM-based wallet belongs to the same channels where the attestation result was generated.
    - Once permission is granted, then a TPM-based wallet sends a download request to the Blockchain peer, who sends this request to the chaincode. Afterwards, the ledger is queried, returning an unencrypted pointer to the contents of that specific block. Those are then forwarded by the peer back to the TPM-based wallet. Knowledge of the pointer would allow a TPM wallet to download the attestation files.

14. In the previous step, we outlined the process that is performed so that a device can obtain data stored in the private ledger. In order to obtain information from the **off-chain storage**, knowledge of the pointer, which indicates the location of the attestation data in the off-chain storage, is required. Based on this pointer, the Blockchain wallet may also get access to the **attestation evidence**, i.e., the encrypted raw traces linked to this attestation report. This process is performed as follows:

    - The device queries the SCB, who will be able to **retrieve and decrypt the pointer** to the data by using its decryption key Key_{SCB}. Then, it can forward the encrypted data to the TPM based-wallet.
    - The Blockchain wallet runs the **Attribute-Based Encryption (ABE)** component to decrypt the block chain data. Since all system traces were encrypted by using our newly designed and **decentralized ABE** scheme leveraging the TPM-based Wallet, the data that will be received by the requesting entity can only be decrypted if its TPM-based Wallet can verify the correctness of the attributes, so that it can create the necessary decryption keys Key_{att}.

Chapter 5

TPM-based Wallet Credentials

5.1 Privacy CA Credential

As described in D4.2 [34], there is a need for a trusted third party that is responsible for attesting and authorizing devices prior to joining the network. In the context of ASSURED, the role of this entity is undertaken by the **Privacy CA**, who is responsible for certifying the correctness of a device prior to verifying it, by generating a registration token that can then be used to request credentials from the Blockchain-related CA. **This is part of the ASSURED Device Registration and Enrollment Phase.**

In the ASSURED use case applications, when it comes to attesting devices with embedded TPM wallets, any ASSURED verifier may want to make sure that the given attestation report was created by a genuine TPM, even if it is unidentified. To meet this requirement, a **Privacy Certificate Authority (CA)** (also called a credential provider) is involved in order to authenticate that the AK holder is a genuine TPM, so that it can afterwards issue a credential to the AK.

Notation	Description
λ, n	security parameters
EK,epk, esk	endorsement key and public/private portion
AK, apk, ask	attestation key and public/private portion
cre	credential of AK
KG	key generation algorithm
aENC/aDEC	asymmetric encryption/decryption operation
sENC/sDEC	symmetric encryption/decryption operation
SIG/VER	signing/verification operation
VER(σ)	signature verification
MAC	message authentication code operation
KDF	key derivation function

Table 5.1: Notation

A TPM AK credential issuing scheme involves three parties: (i) a set of **TPM wallets**, (ii) a set of **devices running the TPM wallets** and (iii) a **credential provider**. The credential provider has a public and secret key pair $(cpk; csk)$, which is used for a signature scheme. Each TPM has a public and secret **Endorsement Key (EK)** pair $(epk; esk)$, which is used for an asymmetric encryption scheme. The Endorsement Key is usually certified by the TPM manufacturer. We

TPM Wallet		Device		CA
$EK = (esk, epk)$		epk, cpk		$ck = (csk, cpk)$, epk
$(ask, apk) \leftarrow \mathsf{KG}(1^\lambda)$	$\xleftarrow{\text{gen}^\dagger AK}$			
	$\xrightarrow{apk}$		$\xrightarrow{epk,apk}$	If apk is not "Restricted" or epk is invalid, return $\perp$, else $s \leftarrow \{0,1\}^n$, $k \leftarrow \{0,1\}^\lambda$ $\mathsf{KE}\|\mathsf{KM} \leftarrow \mathsf{KDF}(s, apk)$ $\hat{s} \leftarrow \mathsf{aENC}_{epk}(s)$ $\hat{k} \leftarrow \mathsf{sENC}_{\mathsf{KE}}(k)$
$s \leftarrow \mathsf{aDEC}_{esk}(\hat{s})$ $\mathsf{KE}\|\mathsf{KM} \leftarrow \mathsf{KDF}(s, apk)$ $k' \leftarrow \mathsf{sDEC}_{\mathsf{KE}}(\hat{k})$ $c \leftarrow \mathsf{MAC}_{\mathsf{KM}}(\hat{k}\|apk)$	$\xleftarrow{\hat{s},\hat{k},\hat{c}}$		$\xleftarrow{\hat{s},\hat{k},\hat{c}}$	$\hat{c} \leftarrow \mathsf{MAC}_{\mathsf{KM}}(\hat{k}\|apk)$
If $c \neq \hat{c}$ return $\perp$, else	$\xrightarrow{k'}$		$\xrightarrow{k'}$	If $k' \neq k$, return $\perp$, else
		If $\mathsf{VER}_{cpk}(cre, apk) = 0$, return $\perp$, else	$\xleftarrow{cre}$	$cre \leftarrow \mathsf{SIG}_{csk}(apk)$
		Store cre		Store $(epk, cpk, apk/cre)$

† Request generating a "Restricted" signing key AK, which can only be used to sign a message or key created by the TPM

Figure 5.1: The Attestation Certificate Authority Solution in TPM 2.0

assume that the credential provider has access to an authentic copy of the public endorsement key and its certificate that is checked before issuing a credential.

The TPM generates a public/secret Attestation Key AK pair (apk, ask), which is used for creating signatures on the traces monitored by an authorized tracer. The TPM attestation key cannot be used unless it is certified by the privacy CA. Thus, whenever a TPM wallet creates an Attestation Key AK, a valid credential from the privacy CA should be issued. This credential clearly certifies that the TPM wallet with an Endorsement Key EK pair $(epk; esk)$ can use its attestation key to create signatures.

A concrete protocol for the privacy CA was introduced in [24], describing the exact communication and message exchange between the TPM and credential provider. The Attestation Certificate Authority Solution involving the privacy CA is shown in Figure 5.1, and the relevant notations used are explained in Table 5.1.

In ASSURED, we further modified this protocol to fit the Blockchain functionality of the ASSURED TPM-based wallet. Specifically, in ASSURED, a DAA key is created in addition to the Attestation Key AK. This DAA key will be used to protect the identity of the TPM Wallet by performing **Direct Anonymous Attestation (DAA)** [16, 23]. The aim of this kind of attestation is to convince a verifier that the device is behaving correctly, without revealing any information about the identity of its TPM wallet. The steps for creating TPM wallet credentials for for the AK and DAA key are described in the enrollment protocol in D4.2 [34], can be summarized as follows:

- To create an Attestation Key (AK), the device running the TPM-based wallet defines a policy

for the key (DefinePolicy). This consists of calculating a policy digest that can be satisfied by any digest that the Security Context Broker has signed. This policy digest represents a valid configuration state that the device needs to be in, so that the usage of the AK is permitted.

- After the attestation key is created using KG algorithm, the TPM-based wallet sends the public part of the attestation key apk to the Privacy CA for verification. This TPM-specific public key holds extra information, such as operational requirements and technical attributes, which will be verified by the Privacy CA.
- After verification that the created AK stems from a genuine TPM wallet with a valid endorsement key certificate, the Certificate Authority needs to authenticate the TPM wallet with an endorsement key EK, i.e., the CA verifies that the TPM wallet holds the secret part of the endorsement key esk before issuing the credential, which is done by the following steps:
  1. The CA chooses random seeds $s \leftarrow \{0,1\}^n$ and $k \leftarrow \{0,1\}^\lambda$.
  2. Then derives the symmetric keys KE and KM from the key derivation function KDF as follows: KE$\|$KM $\leftarrow$ KDF(s, apk).
  3. The CA encrypts the seed s under the TPM wallet public portion of the endorsement key apk as follows: $\hat{s} \leftarrow$ aENC$_{epk}(s)$. This step is required to ensure that only the TPM wallet with the corresponding secret endorsement key esk can recover the seed s.
  4. The privacy CA further uses the symmetric encryption key KE to encrypt the seed k as follows: $\hat{k} \leftarrow$ sENC$_{\mathsf{KE}}(k)$ and calculates $\hat{c} \leftarrow$ MAC$_{\mathsf{KM}}(\hat{k}\|apk)$.
  5. The CA sends $(\hat{s}, \hat{k}, \hat{c})$ to the TPM wallet.
- Upon receiving $(\hat{s}, \hat{k}, \hat{c})$ from the privacy CA, the TPM-based wallet:
  1. Decrypts $\hat{s}$ using its secret endorsement key esk as follows: $s \leftarrow$ aDEC$_{esk}(\hat{s})$.
  2. Recovering the seed s as above enables the TPM wallet to correctly output keys of KDF, KE$\|$KM $\leftarrow$ KDF(s, apk).
  3. Knowing the symmetric decryption key KE, which is the output of KDF, the TPM wallet decrypts $\hat{k}$ as follows: $k' \leftarrow$ sDEC$_{\mathsf{KE}}(\hat{k})$.
  4. The TPM wallet uses the integrity key KM to calculate $c \leftarrow$ MAC$_{\mathsf{KM}}(\hat{k}\|apk)$.
  5. If $c \neq \hat{c}$, the TPM wallet returns $\perp$, else the TPM wallet sends k' to the privacy CA.
- The Privacy CA notifies the SCB about the attributes used by the TPM wallet to create the AK. This notification has a form of a signature, confirming the key contains the right policy and attributes, and the local tracers public key. The SCB generates and authorizes a policy (PolicyPCR) that ensures that the TPM wallet can only execute a signing operation if the device is configured correctly and the tracer authorized it, thus guaranteeing that unauthorized entities cannot use the key.
- To prove the configuration of the device, the TPM wallet executes the PolicyPCR command. If all policies are satisfied correctly then sign command is executed by the TPM wallet. The TPM wallet signature is sent to the Privacy CA that verifies it and creates the credential on the TPM attestation key by using its secret signing key csk to sign the public part of the TPM wallet attestation key apk that originates from the TPM with a certified endorsement key EK as follows $cre \leftarrow$ SIG$_{csk}(apk)$.

- The device leverages its TPM wallet to obtain a **verification ticket**. This ticket proves, in all future contexts, that all previously performed authorizations have been verified at an earlier state and are valid.
- The device verifies that the signature provided on the TPM public attestation key under the CA public key is valid. Finally, the device stores the credential and loads it whenever the TPM wallet wants to generate signatures.
- The CA saves in its records $(epk, cpk, apk/cre)$ that corresponds to the TPM wallet.
- After activating the AK credential, the TPM wallet gets the DAA key that is connected to the AK in the key hierarchy. **The AK will be used to sign the traces measured by an authorized tracer, whereas the DAA key will be used for signing the Attestation Report AR by creating DAA signatures.**
- Finally, the Blockchain keys and the corresponding Blockchain credential are created by the Blockchain CA. The TPM wallet runs TPM2_ActivateCredential to get the Blockchain credential with the associated Blockchain keys.
- The TPM wallet creates Zero Knowledge Proofs using its DAA key that convince a verifier that the TPM wallet has a valid Attestation Key (AK) certified by the Privacy CA, associated with valid Verifiable Credential created by the Blockchain CA. Thus, the TPM-based Blockchain Wallet is linked with the DAA towards the protection of the verifiable credentials and attributes that are required by the devices and users for securely interacting with the Blockchain infrastructure, as show in [27]. The issued VCs and attributes are binded to a specific Wallet, and can only be used if our DAA-Bridge Extension can verify the correctness of the Wallet which, in turn, indicates that the respective (private) key of this VC has not been compromised. The DAA scheme enables a signer to prove the possession of the issued credential to a verifier by providing a signature, which allows the verifier to authenticate the signer without revealing the credential and signer's identity.

5.2 Blockchain CA Credential and its Use by Devices

5.2.1 Setting Up Blockchain based Access Control

If an entity would like to join the ASSURED Blockchain network, it must provide a digital identity which is issued by a trusted Blockchain CA. The Blockchain CA generates the private and public key part for the entity, along with the CA-signed identity. Using Hyperledger Fabric [58] as our main blockchain platform, we can create multiple organizations which in turn may contain multiple peer nodes, admins and clients.

Each organization must contain its own CA for issuing the public and private keys. To this end, an administrator for the organization should first launch a **Fabric CA server**, which the organization will use in order to issue identities. This should be done before any of the peer or orderer nodes of the organization has been deployed. In a second step, an administrator for the organization must be enrolled to become the **CA server's admin**. This operation can be done by using the **Fabric-CA-client**, which can produce the enrollment request to the Blockchain CA server of the organization. Now that the Blockchain CA admin of the organization is enrolled, it can register the identities of the organization's components, such as the peers, orderers, or clients.

It is noted that Attribute-Based Access Control (ABAC) is feasible in Hyperledger Fabric, thanks to the attributes parameter. The attributes can be added to the identity during its registration on the Blockchain CA server.

5.2.2 Registering Devices in the ASSURED Blockchain

If an entity wishes to interact with the ledger, it should have a set of **Verifiable Credentials (VCs)** which are signed by the DAA key of the TPM hosted in the device and are verified by the Blockchain CA. In the context of ASSURED, we use the TPM Wallet of each device to safeguard the private keys of the different entities, working on the principle of hardware based security control when it comes to the safekeeping of sensitive credentials. The various required keys, as well as the verifiable credentials, need to be stored securely, and therefore these are stored in a secure enclave environment which is able to protect them, as well as the attributes of the identity of a device. To this end, it is assumed that each entity needs to interact with the ASSURED ledger through a TPM device. The registration process of a device to the ASSURED blockchain consists of the following steps:

1. In order to get registered to the ASSURED blockchain, the device first requests the TPM Wallet to get registered, and a mutual authentication takes place between the device and the TPM wallet, in order to bind those entities together.
2. Next, the device that hosts the TPM Wallet contacts the Privacy CA to verify the validity of the TPM.
3. The Privacy CA retrieves a **policy digest** from the SCB, which represents the device state for protecting the DAA key usage, and then certifies the TPM of the device, creating the actual DAA key based on the policy digest.
4. The DAA key is then returned to the device, which initiates the DAA JOIN phase, where the TPM Wallet communicates with the Privacy CA to activate the DAA Credentials and verify the attributes of the device.
5. The attributes of the device are stored in the TPM Wallet in the form of **verifiable credentials**.

Using this approach, the DAA key will be stored within the TPM hierarchy key management structure. When the entity would like to extract the key from the TPM, it needs to provide precise attributes to the TPM, so that the TPM can retrieve the valid key to sign a verifiable presentation, as it will be shown in Section 5.2.3.

Once the device attributes are verified, the device can contact the Blockchain CA to ask for registration. This is done by forwarding the signed token which has been provided by the Privacy CA. The Blockchain CA, using the received information, is able to verify the signature and the correctness of the DAA credential, and verify the DAA Key creation and policy protection used. Afterwards, it adds evidence regarding the attributes and proof on the credentials, which are sent back to the device in the state of a Verifiable Credential (VC) that is issued by the blockchain and can be used by the device for its future interactions with the ledger. This chain of operations is shown in Figure 5.2.

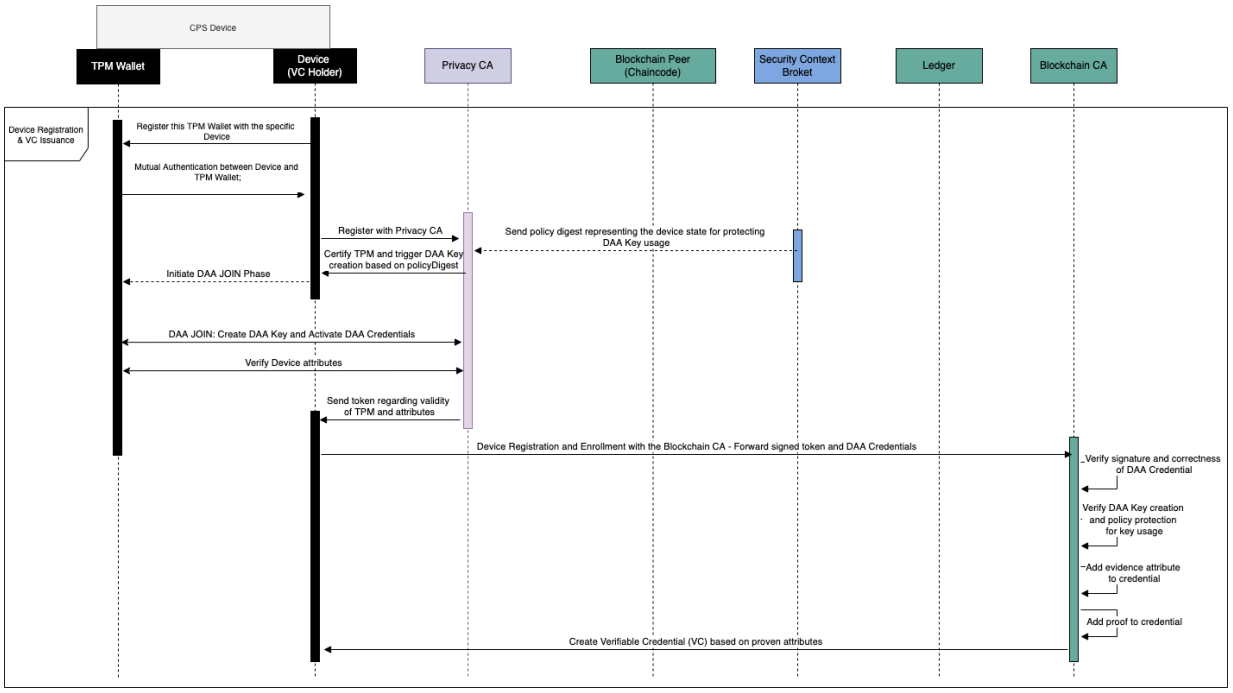

Figure 5.2: Registration of a Device in the ASSURED Blockchain

5.2.3 Controlling Access of Devices to the ASSURED Chaincode

Using the Verifiable Credentials (VC), a TPM wallet generates a **Verifiable Presentation (VP)**, which is signed by the device and is required so that it can query the SCB for providing access to **either the public or the private channels**. As identified in [27], the public channel is supported by a public ledger that stores **encrypted metadata** that corresponds to an attestation report and the **encrypted pointers**. Conversely, in a private channel, the private ledger maintains the **attestation report and its hash**, **unencrypted pointers** and other related data.

In these interactions, access control is concerned with allowing authorized parties (e.g. the devices which host the TPM wallets) to access or place the data collected or generated within the ASSURED Blockchain. As such, edge devices using their TPM wallets can login, connect to the ASSURED Blockchain network and access or place data, **only if they possess the appropriate attributes that allows these actions**. The SCB is responsible for checking whether a device can eventually access the ledger or not, via the retrieved access control policies and the verification of the signed verifiable presentation.

Figure 5.3 showcases this interaction, where an edge device asks to be granted access to a service using its Verifiable Credential.

Based on the flow of actions depicted in Figure 5.3, the flow of actions for controlling access of devices to the ASSURED chaincode is as follows:

1. First, the device seeks to discover the attributes required to engage with a service, and **a call is done in the chaincode**.
2. The Peer, in turn, contacts the SCB, which resolves the request by **retrieving the attributes of the designated service from the Ledger** and returns them to the peer.
3. These attributes are then shared with the device, alongside with a **nonce** generated for this exact request.

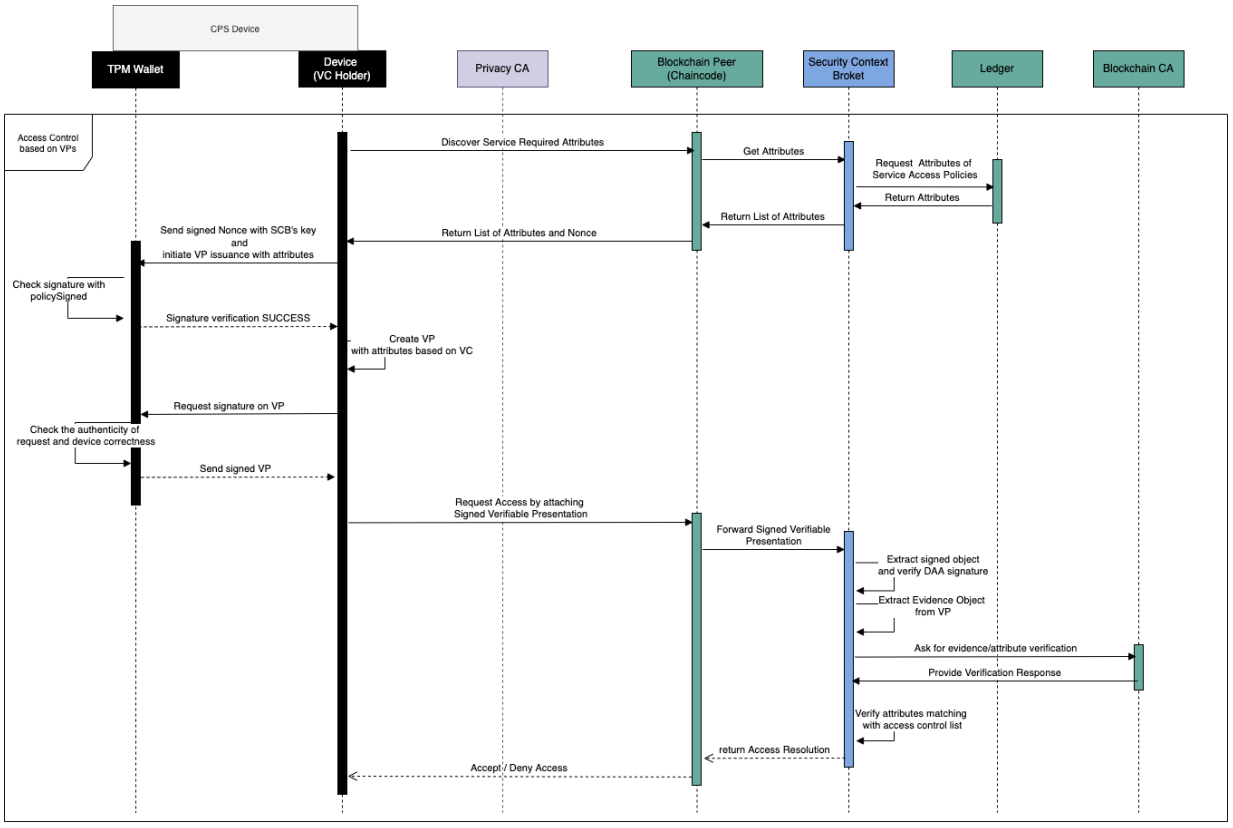

Figure 5.3: Access Request by an Edge Device to a Service using Verifiable Credentials.

4. The device, upon retrieving the requested information, initiates the process for issuing a VP, and **sends the nonce to the TPM Wallet**, which checks and verifies that this signature originates from the SCB.

5. Once the verification is successful, the Device creates the VP with the attributes based on the Verified Credential (VC).

6. This VP is then sent back to the TPM Wallet, which afterwards **authenticates the correctness of the state of the device** and **signs the VP**.

7. At this point, the device holds the necessary information (i.e., the signed VP) that it needs to send to the Blockchain peer to request access to the service. Once this information is retrieved by the peer (e.g. the chaincode), it is **forwarded to the SCB for verification**.

8. The SCB in turn executes a series of steps to **extract the signed object and verify the DAA signature**, and to extract the evidence object out of the VP.

9. Based on this, a call is made to the Blockchain CA to ask for the verification of the evidence provided within the VP.

10. The Blockchain CA responds with a verification message, and finally the SCB performs a check whether the **verified attributes of the device** match with the **access control list of the requested service**.

11. The outcome of this resolution is sent to the Peer, which then allows or rejects the access request that will be sent to the device.

Chapter 6

The TPM Wallet Key Management in ASSURED

6.1 The Need for Secure Key Management in ASSURED

A **key management system with a TPM** is crucial for the ASSURED framework since keys will be used for critical operations, such as encryption, signing, access control management, storage etc. Therefore, it is of paramount importance that the ASSURED architecture provides a standard means of managing the key's lifetime to protect keys and avoid any key-recovery attack. **Key management is the process of managing cryptographic keys within a cryptosystem**. It deals with generating, exchanging, storing, protecting, and replacing keys.

A major challenge that should be addressed by ASSURED is the need to keep cryptographic keys safe and secure. If a single key is compromised, this could lead to a massive data leak, leading to reputation damage and loss of user confidence. For example, if the Attestation Key AK is compromised, then an adversary may use the secret part of the attestation key to create signatures on the system measurements, may also abuse the privacy of the TPM wallet if the key compromised is a DAA key. Thus, ASSURED reliable key management (through the TPM-based Wallet) should establish and specify rules to protect its **confidentiality, integrity, availability, and authentication of source**. The process of key management is essential for all use cases, such as the **Smart Manufacturing** use case, which employs functional-safe sensors/actuators in the manufacturing process, and whose integrity should be protected in order to ensure a safe and secure operational state and avoid harm to machines and human workers.

In the ASSURED vision, there are three expected ways that keys can reside in the TPM-based Wallet, and especially the TPM that provides secure storage and root-of-trust capabilities. These are: i) **Keys generated from a seed**, ii) **keys generated using a random number generator in the TPM**, or iii) **imported keys**. When generating a key, the most important thing that ASSURED use cases have to consider is the security and key protection on the use of the key. For instance, if an inappropriate random number generator is chosen, the resulting generated key will not be secure. Primary keys are generated using a seed that exists in the TPM. The seed used for generating the Endorsement Key EK is associated with the Endorsement hierarchy and cannot be changed.

It is important to note that primary keys are not limited to storage keys. They can also be **asymmetric or symmetric signing keys**. There can be an unlimited number of primary keys of different types (storage, signing) and can refer to different cryptographic algorithms (such as RSA, ECC, SHA-1, SHA-256, etc.). Since there can be a large number of primary keys, it is imprac-

tical to store them all in the TPM Non-Volatile (NV) memory. Therefore, the primary keys are derived from the secret seeds using a **key derivation function (KDF)**, which hashes the primary seed together with a key template. Essentially, the seeds are the actual cryptographic roots. A primary key can be swapped out of the TPM, context-saved, and loaded for the duration of a power cycle, in order to eliminate the time required to regenerate the keys. Since the hierarchies have independent authorization controls (password and policy), they can naturally have separate administrators. Primary keys are created with the command TPM2_CreatePrimary, as is the case of the Attestation Key that is created during the device registration and enrollment phase, as described in D4.2 [34]. When the TPM creates a primary key, it remains on the TPM in the volatile memory. A user may also decide to store the primary key in the persistent memory of the TPM using the TPM2_EvictControl command, which requires the associated hierarchy's authorization. In this case, the key is given a persistent handle that can be later used for accessing the key **??**.

The template for key generation consists of the following parts:

- the first part is a **description of the kind of the generated key**, i.e., whether it is a signing key or an encryption key, asymmetric or symmetric, what type of signing scheme it uses if it is a signing key, the algorithm and key size, etc.
- The second part defines the **entropy of the key**, i.e., the set from where the key is sampled.
- The third part includes **the set of PCRs that should be used to store the configuration state of the device**. This is part of creating the AK during the enrollment process of the device.

A TPM Wallet provides a link between some of its (created) keys which is achieved by the TPM hierarchy. As described in D3.1 [26], the TPM's keys are organised in **key hierarchies**. Each TPM key can be created as a primary key retrieved from a secret seed, or a key created randomly. Keys in a TPM hierarchy have a parent-child relationship. Each hierarchy has a **primary (root) parent keys** and **trees of child keys**. A parent is an encryption key, and a parent key wraps (encrypts) child keys before they leave the TPM. This ensures the creation of a secure boundary in ASSURED when these keys are used in encryption processes, since they provide a layer of protection to the child keys.

A TPM also has a **Non-Volatile (NV) memory** region to store long term keys. Both the **Endorsement Key (EK)** and the **Storage Root Key (SRK)**, which is created by `TPM2_CreatePrimary` command and forms the basis of a key hierarchy that manages secure storage, are stored in the TPM non-volatile memory, as shown in Figure 6.1. Hierarchy can be thought of as having parent-child relationship, or ancestors and descendants. All parent keys are storage keys, which are encryption keys that can wrap (encrypt) their children keys. The storage key protects its children, offering secrecy and integrity when the child key is stored outside the secure hardware boundary of the TPM wallet.

A key can be loaded into the TPM in a wrapped form (encrypted) with a specified parent. Then, the TPM will unwrap it and perform a check along the chain of hierarchy from the parent for authorization. Note that this may require inconvenient password prompts or certain PCR states that might be passed. Since the key is wrapped with another key derived from a hierarchy, it is attached to a hierarchy, but not connected to any parent. For instance, consider the case of **attribute keys**, which are needed in order to access ASSURED Blockchain data by the TPM Wallet that has been encrypted using the ASSURED **Attribute-Based Encryption (ABE)** mechanism [34]. In this case, the TPM Wallet attribute keys should clearly link to its Endorsement Key, that also plays a role of the TPM Wallet identity key.

Loading an attribute key may require authorization with the use of a password. In this case, the problem arises that it may be easier for an attacker to perform an dictionary attack on the password than to compromise the encryption mechanism of the TPM keys, which are wrapped (encrypted) with a strong parent encrypting key. To this end, when a key is loaded in the TPM, it is protected by a mechanism referred to as **dictionary attack protection logic**. Each time an attacker fails to guess the key's authorization, this logic logs the failure. After a configurable number of failures, the TPM blocks further attempts for a configurable amount of time. This limits the speed at which an attacker can try passwords.

Furthermore, authorization is also enabled by checking the current state of the device hosting the TPM Wallet. For instance, the device is authorized to use the AK only if the device's configuration state matches the state that was attested to during registration, which was sent as a *digest list* by the SCB. In a similar context, the use of the loaded Blockchain VCs for creating appropriate verifiable presentations, signed under the DAA Key, is also protected through **policy-based authorization** that a device needs to be at a correct configuration state. We further expand upon this process in Chapter 3.

In the context of the ASSURED framework, in some cases keys are not generated inside the TPM wallets, but may instead be generated by a **trusted authority** and securely shared with the TPM wallets. For instance, a trusted **Blockchain Certification Authority (CA)** creates the Blockchain keys and securely sends them to the TPM wallets. This process is performed by leveraging the capabilities of the ASSURED TPM wallet design by following the following steps:

- The trusted authority creates the key using `TPM2_GetRandom`.
- The trusted authority encrypts the key with the public portion of the TPM endorsement key. This is required in order to associate the key to TPM wallet hierarchy, so that it can be determined which proof value needs to be used.
- The trusted authority signs the encrypted key with its private signing key.
- The encrypted key is sent to the TPM wallet along with a signature that proves that it came from the trusted authority.
- The user verifies the signature on the encrypted key by loading the trusted authority public key. This can be performed with the TPM, by first using the `TPM2_Load` function, and afterwards the `TPM2_VerifySignature` function.
- The user imports the verified, encrypted key into its system using `TPM2_Import`, getting out a loadable, encrypted blob containing the key.
- The user loads the key when the user wishes to use it, using `TPM2_Load`, and uses it as normal.

Keys that might be of interest in the context of ASSURED include the following: (i) **Endorsement Key** - used for securely registering a device in the ASSURED service, (ii) **Attestation Key** – used for the secure enrolment and Zero-Touch Configuration Integrity Verification capabilities of ASSURED. Some may also enhanced with privacy-preserving capabilities such as DAA keys, and (iii) ASSURED **Blockchain keys** – used for protecting the communications between all the edge devices with the backend infrastructure. In the following, we will expand upon the usage of each of these kinds of keys.

6.2 TPM Wallet Key Management

Keys are organised in the TPM in key hierarchies. Except for the leaf keys of a hierarchy, all the other keys serve as parents that protect their children keys. Also, each TPM key can either be created as a **primary key** retrieved from a secret seed, or a **random key** created by using a random number generator. For instance, the TPM Endorsement Key (EK) is a primary key, the TPM Attestation Key (AK) is a random key, and EK is the parent of AK.

Keys can also have restrictions regarding their permitted usage. For example, they can be specified as only **signing** or only **decryption** keys, and they can be restricted to only signing or decrypting certain data, respectively. In general, key attributes determine how the key can be used. The **attribute value** that determines key usage is established at the time of the creation of the key and cannot be changed. An example of a key attribute is the **"Restricted" attribute**, which means that the key can only be used on structures that have a known format. For instance, when a signing key is Restricted, the key can only sign the data created by the TPM itself. Another attribute key designation is **migratable** or **non-migratable**. This key attribute determines whether a key may be transferred from one TPM to another (migratable). Migratable keys are cryptographic keys which are not bound to a specific TPM wallet, and can be generated either inside or outside of a TPM, with suitable authorization.

The following four field are used by the TPM in order to manage the cryptographic keys:

- **Key Handle:** A key handle can be used to identify this key, in the context of the various commands that may use it. Let $k.h$ denote the handle of the key, k.
- **Key Attributes:** Attributes of a key determine how the key can be used. The most notable attributes related to the functionalities of ASSURED are **"fixedTPM"**, **"fixedParent"**, **"Decrypt"**, **"Sign"** and **"Restricted"**. The meanings of the first four attributes are clear from their names. The last one means that the key can only be used on structures that have a known format; e.g., when a signing key is restricted, the key can only sign the data created by the TPM itself. When a restricted signing key is used to certify another key, that certified key must be created by the TPM.
- **Key Public Data and Name:** For an asymmetric key with a public and secret part, denoted as $k = (pk, sk)$, its public data (denoted by $k.p$) includes pk and its attributes and its name (denoted by $k.n$) is a hash value of $k.p$. -
- **Key Blob:** A key stored outside of the TPM is in a format of a key blob that is associated with its parent key PKEY. For an asymmetric key pair $k = (pk, sk)$, the key blob is

$$k.b = (\mathsf{ENC}_{\mathsf{SEK}}(sk), pk, \mathsf{MAC}_{\mathsf{MK}}(\mathsf{ENC}_{\mathsf{SEK}}(sk) \| k.n)),$$

$$(\mathsf{SEK}, \mathsf{MK}) \leftarrow \mathsf{KDF}(\mathsf{PKEY}, \mathsf{SALT}),$$

  where SEK and MK are a symmetric encryption key and a MAC key respectively; SALT is a data string used to make PKEY reusable.

6.3 TPM Endorsement Key & Verifiable Credentials

Each TPM must have an **Endorsement Key (EK)** embedded in it. The Endorsement Key represents the TPM's identity in the ASSURED framework, and this key is usually certified by the TPM

Manufacturer. TPM Manufacturers usually provide the Endorsement Key pair and store it in the tamper resistant non-volatile memory before shipping the TPM. **The EK is an asymmetric encryption key, and is never used to generate signatures.** The Endorsement Key pair consists of the private key which is embedded in the TPM and is never shared with any other parties or components. The public key is contained in a certificate and is only used for encrypting data.

The TPM EK is an asymmetric (public) encryption and (private) decryption key pair, rather than a (private) signing and (public) verification key pair. However, in order to perform attestation operations, the TPM uses its Attestation Key (AK) instead of its Endorsement Key. However, in order to prove to a potential credential provider that a certified EK and a given AK belong to the same TPM, it is essential to implement a method in order to achieve a *cryptographic binding between a TPM's AK and EK.* This binding is achieved as follows:

1. Given the public portions of a certified EK and an unauthenticated AK, the credential provider encrypts a challenge, and then "wraps" the challenge encryption key together with the public key of the AK, under the public key of the EK of the TPM Wallet.
2. The encrypted challenge and the wrapped keys are then delivered to the TPM Wallet, which loads the EK and AK onto the TPM and asks the TPM to return the challenge encryption key. This TPM operation is called "activate credential".
3. The TPM will first decrypt the wrapped keys using the private key of the EK and then validate that both the public and private portions of the AK are loaded on the TPM.
4. If this validation is successful, the TPM will return the challenge symmetric encryption key, that will be used by the device to decrypt the challenge.
5. After the challenge response is sent, the credential provider creates a credential to the AK and makes it available to the device.

Thus, a valid TPM credential is created that clearly links the TPM Wallet Attestation key with its Endorsement Key [34]. If the TPM Wallet wants to join the ASSURED Blockchain services, the Blockchain CA checks the validity of the TPM Wallet through the token provided by the Privacy CA that certifies and activates the TPM credentials, and then issues **Verifiable Credentials (VCs)** based on the attributes proven by the device. These VCs are linked to the TPM Wallet Endorsement Key, and are also protected through the creation of an appropriate policy that binds the usage of this VC to specific PCR values, which represent the correct state of the device that needs to create and sign **Verifiable Presentations (VPs)**.

6.4 TPM Attestation Keys

As analyzed in Section 6.3, the EK is an encryption/decryption key that cannot be used for signing purposes, but it can be used to **wrap (i.e., encrypt) other keys**. Therefore, there is a need to create a **signing attestation key (AK)** associated with the endorsement hierarchy. Thus, the AK used is encrypted (wrapped) by the EK. The TPM wallet can only load the AK under its unique EK and use it to sign data.

The Attestation Key (AK) can be seen as a pseudo-identity key for the TPM. A device can have multiple AKs for different purposes rather than combining all keys into one key and exposing extra information when proving one of their properties. For instance, a device can have an AK to sign

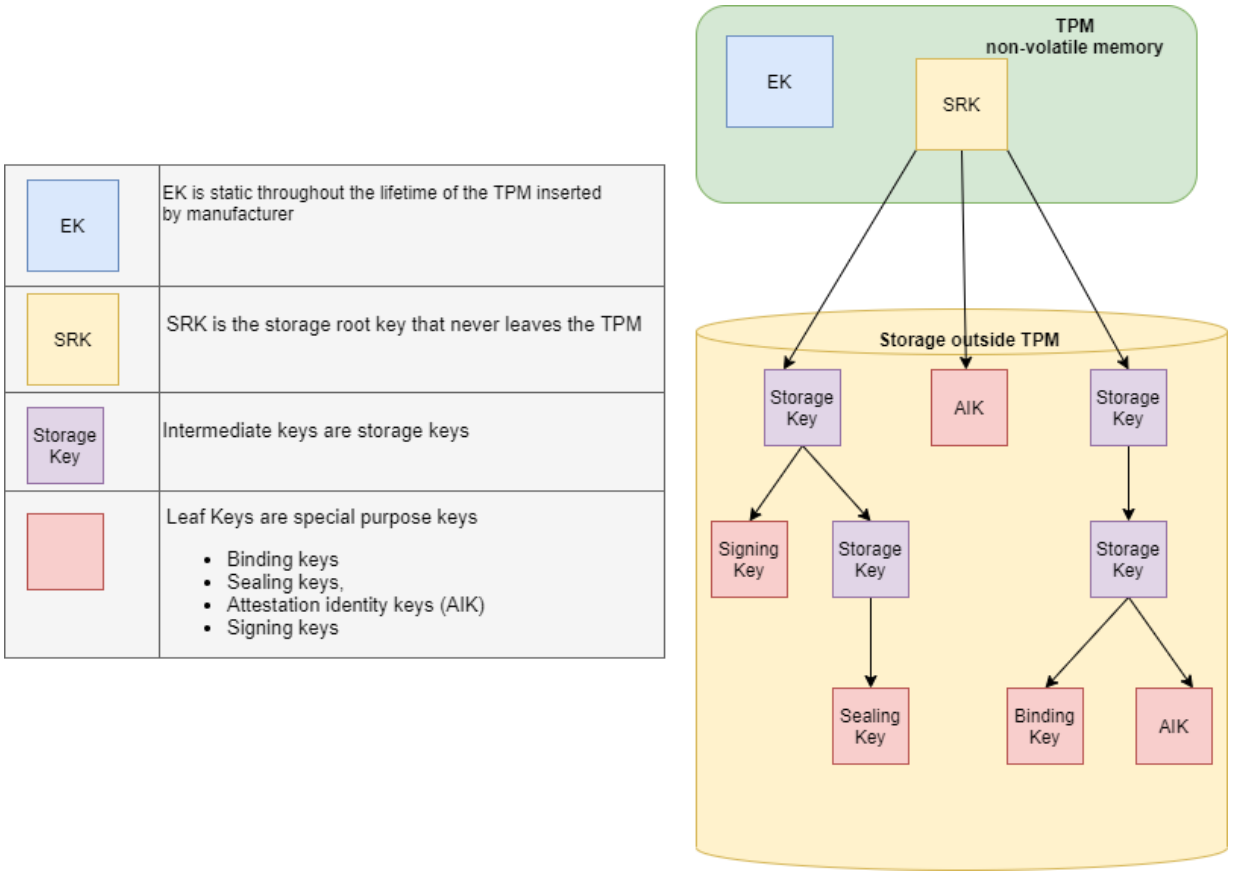

Figure 6.1: TPM Key Hierarchy

traces monitored by an authenticated tracer, and another attestation key holding some attributes that allows specific Blockchain functionalities.

Moreover, a device can have a **DAA key**, that enables it to perform attestation tasks in a **privacy preserving manner**. This kind of attestation is referred to as **direct anonymous attestation (DAA)**, which is based on group signatures and provides a complex method for proving that an attestation key was created by a TPM, without disclosing information as to which TPM created it. The advantage of this protocol is that it lets the AK creator choose a variable amount of knowledge that the privacy CA should be permitted to access, ranging from perfect anonymity (i.e., when a certificate is created, the privacy CA is given proof that an AK belongs to an unidentified TPM) to perfect knowledge (i.e., the privacy CA knows which EK is associated with an AK when providing a pseudonymous certificate for the AK, as described in Section 5.2).

During device enrollment, the Privacy CA requests and verifies the creation of a restrained asymmetric Attestation Key (AK) pair on a device's trusted security anchor (TPM), where the use of the AK is certified to require that a specified selection of TPM Platform Configuration Registers (PCRs) contain authorized aggregated Trusted Reference Values (TRVs), thus ensuring that the device's secret AK (ask) can be used only if that device is in a correct (authorized) state. This is based on the **policy-protected key usage** attribute described in D3.1 [26] as part of the overall ASSURED secure key management landscape.

As it was also analyzed in D3.2 [32], the enrollment protocol consists of the following steps:

1. The protocol begins locally on the SCB, where a policy digest is computed over the Command Code (CC) of **TPM2_PolicyAuthorize** and the name of SCB's EK. Note that such

policies are called *flexible*, since any object ϕ bound to the policy can only be used in a policy session with the TPM after fulfilling some policy (e.g., that the PCRs are in a particular state) which the policy's owner (SCB in our case) has authorized (signed).

2. The policy digest, together with a template describing the key's characteristic traits (e.g., attributes and type), is then sent to device. Besides producing and returning the AK object, the TPM also returns a signed ticket over the object to denote that it was created inside the TPM.

3. This "creation" ticket, together with the newly created AK object and EK, are then passed to TPM2_CertifyCreation, where the TPM vouches that it was involved in producing AK (if the ticket holds) by signing the AK object along with some internal state information(Check Section 5.2.

4. Then, AK is stored persistently in TPM NV memory (using **TPM2_EvictControl**).

5. Finally, the TPM wallet presents the AK and certificate to the SCB, who verifies the certificate's signature and scrutinizes its details to ascertain that the AK was created legitimately. If everything holds, the TPM wallet is permitted to enter the network.

6.5 TPM Blockchain Access Keys

In ASSURED, each of the Blockchain users and/or devices should be issued a **Blockchain credential**, including a **Blockchain key pair**, i.e., both the private and the public part of the key that essentially represents the **Verifiable Credentials (VCs** to be issued to the user/device. The public key and the corresponding attributes (as well as the device identity and the TPM identity through its EK) are constructed into a Blockchain VC, while the private key is sent to the user through the a secure and authentic communication channel, *by using key templates encrypted under the seed which can then be re-created only by the TPM that has access to a specific EK*. This process has also been described in D4.2 [34].

This key pair is generated by the Blockchain CA during the **user enrollment** of the device to the Blockchain network. After obtaining the VC, the device, which is equipped with the TPM, will store it as part of its TPM Wallet. Inside the TPM, the key is stored securely while the Wallet enables the storage of the VC. The latter can only be used by authorized processes, since any self-issued Verifiable Presentation (based on this VC) should be signed under the DAA Key in order to be valid. In turn, the DAA key is also binded to specific process running inside the device.

While needed, the user will **request the TPM to output its key pair**, and then the TPM will decrypt and output the key pair if the user's **attribute authentication is valid**. Recall that this authentication process is performed with the ABAC mechanism, through the construction of **Verifiable Presentation (VP)** of the device attributes by the TPM, as it has been described in Section 4.1.2. The user is afterwards able to use the output key pair to perform related **Blockchain operations**. For example, a client can use the output private key to **sign a transaction** which will be sent to a peer, and the peer can later use its own private key output by the TPM to sign a transaction execution result for the client. The above process is applicable to all Blockchain users. Recall that these can be categorized into **peers** and **orderers**, but since the orderers can be seen as a type of peers, the process is similar in both cases.

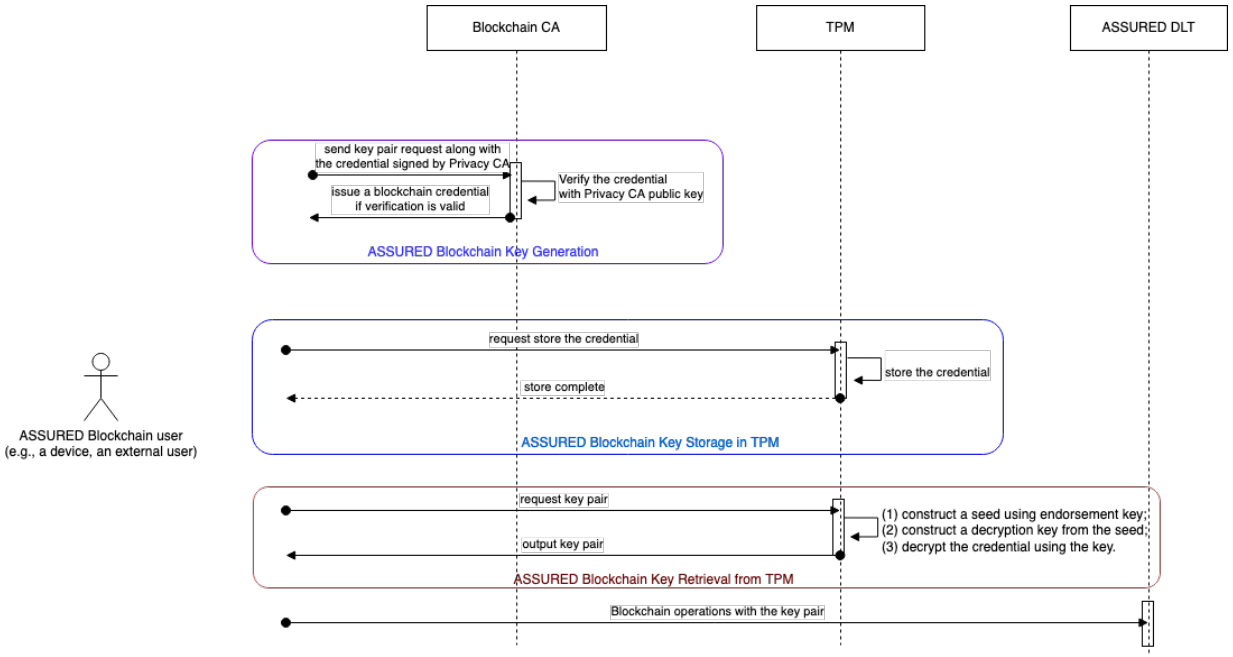

Figure 6.2: Blockchain Key Management in TPM

A similar process can also be applied when a peer needs to **store its key pair**, as well as to store the **certificates, i.e., the public keys, of other clients** who are registered within the same channel. This process consists of the following steps:

1. First, the peer should already have the Blockchain CA certificate stored in its TPM.
2. Once receiving a certificate sent by a client, the peer will request the TPM to output the Blockchain CA's certificate - with the corresponding public key, so that the peer can verify if the client's certificate (with public key) is signed correctly by the CA.
3. If the client's certificate is correctly signed, this certificate will be stored within the TPM and encrypted by the TPM Endorsement Key.

Note that the above process is also applicable to the procedure of storing other peers'/orderers' certificates into the peer's TPM.

Chapter 7

Anonymous Secure Channel Establishment

As described above, one of the key functionalities that is supported by the ASSURED TPM-based Wallet is **key management** when it comes to the **creation, usage and safeguarding of the various crypto materials** needed for supporting the secure participation of a device in the target "Systems-of-Systems" environment. In this context, core key materials pertain to guaranteeing the **secure operation and integration of the attestation tasks and evidence** (i.e., Attestation Key), respectively, as well as the **continuous authentication and authorization of participating devices and users** (i.e., DAA Key, private and public part of the key based on the issued Blockchain Verifiable Credentials) ensuring that messages originate from genuine devices without making individual devices traceable throughout the system.

For the latter, as described in Chapter 3, the ASSURED TPM-based Wallet provides also strong levels of assurance on the Verifiable Credentials' (VC) origin and integrity in order to be able to decide that credentials can be trusted (in a transparent and automated manner) and that they really belong to the claimed entity exhibiting configuration and run-time behavioral correctness before being allowed to interact with the DLT (through the Blockchain Peer [27, 35]). Towards this direction, ASSURED **enables the use of crypto operations of DAA (ECC, Blind Signatures, Zero Knowledge Proofs), generating hw-based keys to be binded to a device's VC (Wallet) that can provide verifiable evidence and assurances about both the presented VC's origin and integrity but also any other attributes required to be presented by the device for accessing a service of the Blockchain (through appropriate self-issued Verifiable Presentations (VPs)); e.g., requesting access the attestation history of a specific device**.

However, one of the main challenges in the integration of the ASSURED Enhanced DAA protocol (enabling also revocation capabilities [32]), for enabling and protecting the self-issuance and usage of such VPs, **is the assumption of the existence of a secure communication channel during the $SETUP$ and $JOIN$ phases**. This basically puts a significant constraint to the device enrollment process [34] since it needs to execute over a secure channel for the DAA and Attestation Keys certification as well as the secure exchange of the *attestation policies* needed by the Configuration Integrity Verification scheme [32].

Compounding this issue, in ASSURED we are planning to enhance this DAA protocol and try to solve the assumption that the Signer and the Issuer have established a one-way authentic channel [12], i.e., *anonymous and secure exchange of the parameters for the pairings that are executed for the creation of the ECC key, and the activate credentials step*. **This will also enable the creation of (ephemeral) symmetric keys for providing strong confidentiality guarantees**

during the data communication and exchange between two devices or between a device and the backend Blockchain infrastructure. *Recall that currently all attestation-related keys are essentially restrictive signing keys used for protecting the integrity of exchanged data*.

7.1 Key Exchange with Anonymous Authentication

One trivial solution would be the establisment and usage of TLS communication channels. Transport Layer Security, previously called Secure Socket Layer, is a cryptographic protocol tailored to ensure security, data integrity, trustworthiness, malware prevention and granular control of transmitted data over the the Internet. In short, it provides authentication, confidentiality, and integrity. The latest version is TLS 1.3 [55]. TLS 1.3 starts with a key exchange phase where the client and the server choose a random number r (256 bits fresh nonce), a list of favored symmetric cipher-suites, and an ephemeral (EC)DH private key. The private key belongs to one TLS DH group. The client is allowed to send several keys and the server takes up the responsibility to pick the group to continue. **After this phase the keying material, cryptographic parameter, and an encrypted channel are generated and established.**

The next phase is the **authentication phase**. The server sends its certificate (X.509), with appropriate credentials/tokens that can be used for its efficient verification (signature over the transcript of the entire handshake messages using the private key), encrypts the extensions and terminates (HMAC over the transcript and the derived key). The client replies with “Secure ACK” to the server. All messages in the subsequent communication are now protected by AEAD with the derived key; this constitutes the third phase [55].

As with almost every protocol, TLS has some drawbacks when it comes to its applicability to various application domains other than HTTP and even with HTTP. Most noteworthy are high **latency, Men in the Middle (MiM) Attacks, network complexity, platform support, and implementation costs when using a certificate that is not free available**. Due to the widespread of TLS, it is a common attack surface to break the integrity and security. Here, ordinary attacks are Ciphersuite rollback attack [62], change chipher spec dropping attack in [62], version rollback attack [62], RSA-based sessions attack [44], adaptive chosen ciphertext attack on PKCS#1 [10], timing attacks [52], Cross-protocol attack based on ECC key exchange [51], SKIP-TLS attack [9], Factoring Attack on RSA-EXPORT Keys attack [2], Logjam attack [6], etc.

7.2 Key Exchange with Anonymous Authentication

In ASSURED, our protocol aims to be more lightweight than TLS as well as providing different levels of *device- or user-controlled anonymity*. **We are planning to use an ephemeral/static ECDH key agreement protocol with anonymous authentication between a Prover (P) and a Verifier (V)**. This is based on the notation from Chen et al. [23] which also leverages the cryptographic primitives used in the DAA protocol.

The Prover has a Camenisch-Lysyanskaya credential [16] and a private key. She randomizes the credential and sends it to the Verifier who in turn verifies the randomized points. The Verifier then calculates a random point and the shared secret which is then sent back to the Prover. This parameter is subsequently used for calculating the same shared secret as the Verifier.

Both parties have now agreed on a common shared secret which can be used for deriving symmetric session keys KMAC, i.e, after receiving a message with KMAC integrity from the Prover

(P), the Verifier (F) can assume that P has the private key belonging to the ticket.

An early draft of this described protocol is shown in figure 7.1[1]. On a successful run of this key agreement process, both parties know a common shared secret but they are not aware of any other identifying information of the other entity; thus, achieving **anonymous node authentication** which is a core requirement in SoS environments and especially in safety-critical applications as the ones (for example) envisaged in the defined use cases (e.g., Public Safety). **This property will allow the (on-demand) privacy-preserving enrollment of the devices in a network**, thus, leading to an enhanced DAA $JOIN$ phases where a node (together with the underlying TPM) can verify the validity of its root of trust and unique device identity without revealing the actual identity of the attached TPM (i.e., certification of endorsement key in a privacy-preserving manner). Furthermore, a third party (e.g an attacker) can neither learn the shared secret, nor information on the identity of the parties.

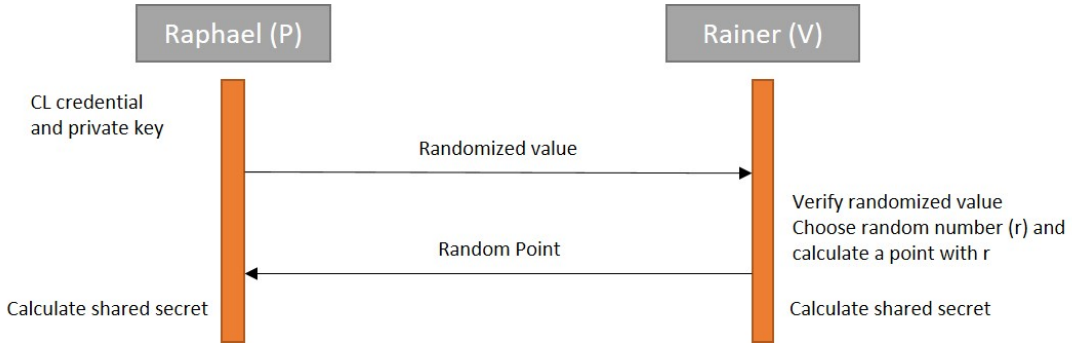

Figure 7.1: ADHKE - Anonymous Diffie-Hellman Key Exchange

The proposed scheme is expected to offer a number of benefits, such as **no dedicated HW-TPM is required**, can be implemented in IoT Devices that offer interfaces that are not compatible with trusted components (e.g., Java card), **improved performance** and **privacy** in relation to TLS.

7.3 Research Plan for Establishing Ephemeral Keys with Diffie-Hellman Key Agreement Protocol & Advanced ASSURED Key Management

The security mechanisms proposed for the ASSURED platform, when it comes to the establishment of such ephemeral keys are of different degrees of maturity. On one hand, TPMs are well-established with a large research body and a number of commercial vendors offering such devices and solutions. On the other hand, the proposed Ephemeral Diffie-Hellman Key Agreement Protocol is a scheme that will be defined in the second release of the ASSURED TPM-based Wallet.

For the automated, anonymous establishment of DH-Keys in ASSURED, the following design and implementation plan will be followed:

1. Establishment of the mathematical scheme in detail, e.g., definition of protocol roles and actions, pre-conditions and security claims and assumptions.

[1] The finalization of the concrete models and implementation aspects will be provided in the context of D4.6

2. Evaluation of the protocol axioms, security analysis and discussion in the ASSURED trust model [26].
3. Implementation of the scheme within the ASSURED TPM-based Wallet so as to be able to support the establishment of secure and authentic communication channels between all actors in the ASSURED ecosystem.
4. Classification of the resource and performance measurements coupled with a detailed evaluation.
5. Study of the applicability of hardware-based cryptography accelerators.
6. Further research on the extension of the basic scheme towards a direct pseudonymous key agreement protocol and a direct pseudonymous encryption/decryption scheme.

7.4 Research Plan for Leveraging the Key Management Functionalities of the TPM in a Property-Preserving Manner

As it has been highlighted throughout this deliverable, there are several types of keys stored separately within the TPM storage. However, TPM 2.0 provides a **tree-based hierarchical structure**, where a parent key acts as the root, which subsequently wraps and encrypts all of its children keys. Specifically, the TPM specification provides the following hierarchies for key management:

- **Endorsement:** The endorsement hierarchy constitutes a privacy-sensitive tree, and is the hierarchy of choice when the user is subject to privacy concerns. TPM and platform vendors certify that, in this hierarchy, primary keys are constrained to an authentic TPM attached to an authentic platform.
- **Owner:** Also referred to as **storage hierarchy**, this is intended to be used by the platform or device owner.
- **Platform:** Intended to be used by the platform manufacturer.
- **Null:** An ephemeral hierarchy, where the created keys will not be stored in case of a system reboot.

In the context of ASSURED, future research could include an investigation into the potential usage of the hierarchical capabilities provided by the TPM, in order to perform the key management process in a more efficient manner.

However, the main challenge in this case is to use these features of the TPM, while simultaneously **maintaining the properties of the stored keys**. For example, since the DAA key is intended to grant privacy preserving properties to the attestation process, it should be characterized by **unlinkability** and **untraceability**. Therefore, the question arises: *Is it possible to leverage the key management capabilities of the TPM in order to link the DAA key with an Attestation Key (AK), without compromising the desired properties of the DAA key?* This question should be addressed, if such a key management scheme is implemented in the context of ASSURED.

Chapter 8

Conclusion

In this deliverable, we provided descriptions for the main TPM wallet functionalities within the ASSURED Blockchain services that include the following:

- Providing strong user authentication in an anonymous way through a lightweight protocol, that provides different levels of *device- or user-controlled anonymity*.
- Controlling and authorizing access to private or public ledger channels based on the authentication process, e.g., to authorize access to operations on different ledgers. This is achieved through the **Verifiable Credentials** that are binded to the TPM wallet DAA keys. Generating DAA signatures on the Verifiable Credentials that were issued by the Blockchain CA during the enrollment phase can provide verifiable evidence and assurances about both the presented VC's origin and integrity, but also any other attributes required to be presented by the device for accessing a service of the Blockchain.
- Guaranteeing the **secure operation and integration of the attestation tasks and evidence** (i.e., Attestation Key), respectively, as well as the **continuous authentication and authorization of participating devices and users** that allows continuously attesting and assessing the security of all involved devices in a privacy-preserving manner through DAA signatures on the Verifiable Credentials, ensuring that messages originate from genuine devices, without making individual devices traceable throughout the system.
- Securely storing the credentials and keys of the devices, based on the TPM's secure key storage, and efficiently verifying Blockchain updates and be able to handle basic ledger operation fees and digital transactions.

We will expand on the description of the TPM-based Wallet in the next version of this deliverable [36], where further details regarding its structure and implementation will be provided.

List of Abbreviations

Abbreviation	Translation
ABAC	Attribute-based Access Control
AE	Authenticated Encryption
ABE	Attribute-based Encryption
AK	Attestation Key
CA	Certification Authority
CFA	Control-flow Attestation
CIV	Configuration Integrity Verification
CSR	Certificate Signing Request
DAA	Direct Anonymous Attestation
DLT	Distributed Ledger technology
EA	Enhanced Authorization
EK	Endorsement Key
GSS	Ground Station Server
MSPL	Medium-level Security Policy Language (MSPL)
NMS	Network Management System
Privacy CA	Privacy Certification Authority
Prv	Prover
PCR	Platform Configuration Register
PLC	Program Logic Controller
RA	Risk Assessment
RAT	Remote Attestation
SCB	Security Context Broker
SoS	Systems of Systems
SSR	Secure Server Router
S-ZTP	Secure Zero Touch provisioning
TC	Trusted Component
TLS	Transport Layer Security
TPM	Trusted Platform Module
VC	Verifiable Credential
Vf	Virtual Function
VM	Virtual Machine
VP	Verifiable Presentation
Vrf	Verifier
WP	Work Package
ZTP	Zero Touch Provisioning

References

[1] Privacy-enhancing identity management. *Information Security Technical Report*, 9(1):35 – 44, 2004.

[2] Factoring rsa export keys - freak (cve-2015-0204). https://access.redhat.com/blogs/766093/posts/1976563, 2015. Accessed: 2020-12-01.

[3] Trusted platform module (tpm). https://trustedcomputinggroup.org/work-groups/trusted-platform-module, 2020. Accessed: 2020-12-01.

[4] Digital Identity: Leveraging the SSI Concept to Build Trust. ENISA Report, European Union Agency for Cybersecurity (ENISA), 2022.

[5] C. Adams, S. Farrell, T. Kause, and T. Mononen. X. 509 Public Key Infrastructure Certificate Management Protocol (CMP), 2005. http://www.hjp.at/doc/rfc/rfc4210.html.

[6] David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella-Béguelin, and Paul Zimmermann. Imperfect forward secrecy. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, oct 2015.

[7] Man Ho Au, Willy Susilo, and Yi Mu. Constant-Size Dynamic k-TAA. In *Proceedings of the 5th International Conference on Security and Cryptography for Networks (SCN'06)*, page 111–125, 2006.

[8] Zinaida Benenson, Anna Girard, and Ioannis Krontiris. User Acceptance Factors for Anonymous Credentials: An Empirical Investigation. In *Workshop on the Economics of Information Security (WEIS)*, 2015.

[9] Benjamin Beurdouche, Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cedric Fournet, Markulf Kohlweiss, Alfredo Pironti, Pierre-Yves Strub, and Jean Karim Zinzindohoue. A messy state of the union: Taming the composite state machines of TLS. In *2015 IEEE Symposium on Security and Privacy*. IEEE, may 2015.

[10] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In *Advances in Cryptology — CRYPTO '98*, pages 1–12. Springer Berlin Heidelberg, 1998.

[11] Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct Anonymous Attestation. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, page 132–145, 2004.

[12] Ernie Brickell, Liqun Chen, and Jiangtao Li. Simplified security notions of direct anonymous attestation and a concrete scheme from pairings. *International Journal of Information Security*, 8(5):315–330, feb 2009.

[13] Jan Camenisch, Manu Drijvers, and Anja Lehmann. Universally composable direct anonymous attestation. In *Public-Key Cryptography – PKC 2016*, volume 9615 of *LNCS*, pages 234–264. Springer, 2016.

[14] Jan Camenisch and Anna Lysyanskaya. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In *Advances in Cryptology — EUROCRYPT 2001*, pages 93–118, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[15] Jan Camenisch and Anna Lysyanskaya. A Signature Scheme with Efficient Protocols. In *Proceedings of the 3rd International Conference on Security in Communication Networks (SCN'02)*, page 268–289, 2002.

[16] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology – CRYPTO 2004*, pages 56–72. Springer Berlin Heidelberg, 2004.

[17] Jan Camenisch and Els Van Herreweghen. Design and Implementation of the Idemix Anonymous Credential System. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS)*, page 21–30, 2002.

[18] Kim Cameron, Reinhard Posch, and Kai Rannenberg. Proposal for a Common Identity Framework: A User-Centric Identity Metasystem. In K. Rannenberg, D. Royer, and A. Deuker, editors, *The Future of Identity in the Information Society*, page 477–500, 2009.

[19] David W. Chadwick. *Federated Identity Management*, pages 96–120. 2009.

[20] David Chaum. Blind Signatures for Untraceable Payments. In *Advances in Cryptology: Proceedings of CRYPTO '82*, pages 199–203. Plenum, 1982.

[21] David Chaum and Eugène van Heyst. Group Signatures. In *Advances in Cryptology — EUROCRYPT '91*, pages 257–265, 1991.

[22] David L. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–90, February 1981.

[23] Liqun Chen, Dan Page, and Nigel P. Smart. On the design and implementation of an efficient DAA scheme. In *Lecture Notes in Computer Science*, pages 223–237. Springer Berlin Heidelberg, 2010.

[24] Liqun Chen and Bogdan Warinschi. Security of the tcg privacy-ca solution. In *2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, pages 609–616. IEEE, 2010.

[25] Sebastian Clauß and Marit Köhntopp. Identity Management and Its Support of Multilateral Security. *Computer Networks*, 37(2):205–219, August 2001.

[26] The ASSURED Consortium. Assured attestation model & specification. Deliverable D3.1, November 2021.

[27] The ASSURED Consortium. Assured blockchain architecture. Deliverable D4.1, November 2021.

[28] The ASSURED Consortium. Assured blockchain architecture. Deliverable D1.4, November 2021.

[29] The ASSURED Consortium. Assured reference architecture. Deliverable D1.2, May 2021.

[30] The ASSURED Consortium. Assured use cases & security requirements. Deliverable D1.1, February 2021.

[31] The ASSURED Consortium. Assured blockchain-based control services and crypto functions for decentralized data storage, sharing and access control. Deliverable D4.3, February 2022.

[32] The ASSURED Consortium. Assured layered attestation and runtime verification enablers design implementation. Deliverable D3.2, November 2022.

[33] The ASSURED Consortium. Assured secure and scalable aggregate network attestation. Deliverable D3.6, February 2022.

[34] The ASSURED Consortium. Assured secure distributed ledger maintenance & data management. Deliverable D4.2, February 2022.

[35] The ASSURED Consortium. Security context broker specification and smart contract definition & implementation for policy enforcement. Deliverable D2.2, February 2022.

[36] The ASSURED Consortium. Assured tc-based functionalities - version 2. Deliverable D4.6, February 2023.

[37] Heini Bergsson Debes and Thanassis Giannetsos. Segregating keys from noncense: Timely exfil of ephemeral keys from embedded systems. In *2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 92–101, 2021.

[38] DIF. Decentralized Identity Foundation, 2020. https://identity.foundation/.

[39] ESSIF. Working for development, integration and adoption of Self-Sovereign Identities (SSI) technologies, 2020. https://essif-lab.eu.

[40] essif-lab functional architecture. https://essif-lab.github.io/framework/docs/essifLab-fw-func-arch, 2022.

[41] essif-lab vision. https://essif-lab.github.io/framework/docs/essifLab-vision, 2022.

[42] Simone Fischer-Hübner, Chris Hoofnagle, Ioannis Krontiris, Kai Rannenberg, and Michael Waidner. Online Privacy: Towards Informational Self-Determination on the Internet (Dagstuhl Perspectives Workshop 11061). *Dagstuhl Manifestos*, 1:1–20, 01 2011.

[43] ISO/IEC JTC 1/SC 27. IT Security and Privacy — A framework for identity management — Part 1: Terminology and concepts. ISO/IEC 24760-1:2019, 2019.

[44] Vlastimil Klíma, Ondrej Pokorný, and Tomáš Rosa. Attacking RSA-based sessions in SSL/TLS. In *Lecture Notes in Computer Science*, pages 426–440. Springer Berlin Heidelberg, 2003.

[45] Ioannis Krontiris, Zinaida Benenson, Anna Girard, Ahmad Sabouri, Kai Rannenberg, and Peter Schoo. Privacy-ABCs as a Case for Studying the Adoption of PETs by Users and Service Providers. In *Privacy Technologies and Policy*, pages 104–123. Springer, 2016.

[46] Benjamin Larsen, Heini Bergsson Debes, and Thanassis Giannetsos. Cloudvaults: Integrating trust extensions into system integrity verification for cloud-based environments. In *Computer Security*, pages 197–220, Cham, 2020. Springer International Publishing.

[47] Benjamin Larsen, Thanassis Giannetsos, Ioannis Krontiris, and Kenneth Goldman. Direct anonymous attestation on the road: Efficient and privacy-preserving revocation in c-its. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '21, page 48–59, New York, NY, USA, 2021.

[48] Loïc Lesavre, Priam Varin, Peter Mell, Michael Davidson, and James Shook. A Taxonomic Approach to Understanding Emerging Blockchain Identity Management Systems, Jan. 2020. NIST Cybersecurity White Paper. Available online at https://doi.org/10.6028/NIST.CSWP.01142020.

[49] Tobias Looker and Orie Steele. BBS+ Signatures 2020, 2020. https://w3c-ccg.github.io/ldp-bbs2020/.

[50] E. Maler and D. Reed. The Venn of Identity: Options and Issues in Federated Identity Management. *IEEE Security Privacy*, 6(2):16–23, 2008.

[51] Nikos Mavrogiannopoulos, Frederik Vercauteren, Vesselin Velichkov, and Bart Preneel. A cross-protocol attack on the TLS protocol. In *Proceedings of the 2012 ACM conference on Computer and communications security - CCS '12*. ACM Press, 2012.

[52] Christopher Meyer and Jörg Schwenk. SoK: Lessons learned from SSL/TLS attacks. In *Information Security Applications*, pages 189–209. Springer International Publishing, 2014.

[53] Christian Paquin and Greg Zaverucha. U-Prove Cryptographic Specification V1.1. Technical report, December 2013.

[54] Kai Rannenberg, Jan Camenisch, and Ahmad Sabouri, editors. *Attribute-based Credentials for Trust: Identity in the Information Society*. Springer, 2015.

[55] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, August 2018.

[56] Ahmad Sabouri, Ioannis Krontiris, and Kai Rannenberg. Trust relationships in privacy-ABCs ecosystems. In *International Conference on Trust, Privacy and Security in Digital Business*, pages 13–23. Springer International Publishing, 2014.

[57] Trusted network connect (tnc) howto. https://wiki.strongswan.org/projects/1/wiki/trustednetworkconnect, 2018.

[58] Hyperledger fabric. https://www.hyperledger.org/use/fabric, 2022.

[59] Trusted Computing Group. TPM Main Part 1 Design Principles, Specification Version 1.2. http://www.trustedcomputinggroup.org.

[60] W3C Recommendation. Verifiable Credentials Data Model 1.0, 2019. https://www.w3.org/TR/2019/REC-vc-data-model-20191119/.

[61] W3C Recommendation. Decentralized Identifiers (DIDs) v1.0 Core architecture, data model, and representations, 2021. https://www.w3.org/TR/did-core/.

[62] David Wagner and Bruce Schneier. Analysis of the ssl 3.0 protocol. In *Proceedings of the 2nd Conference on Proceedings of the Second USENIX Workshop on Electronic Commerce - Volume 2*, WOEC'96, page 4, USA, 1996. USENIX Association.

[63] Xuanxia Yao, Zhi Chen, and Ye Tian. A lightweight attribute-based encryption scheme for the internet of things. *Future Generation Computer Systems*, 49:104–112, 2015.