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

D4.3: ASSURED SECURE DISTRIBUTED LEDGER MAINTENANCE & DATA MANAGEMENT

Revision: v.1.0

Work package	WP 4
Task	Task 4.2
Due date	28/02/2022
Deliverable lead	SURREY
Version	1.0
Authors	Nada El Kassem (SURREY)
Reviewers	Liqun Chen (SURREY) Dimitris Papamartzivanos (UBITECH)
Abstract	D4.2 proceeds with the description of all cryptographic primitives and protocols for safeguarding the secure communication (between devices) as well as the secure and privacy-preserving data sharing over the ASSURED Blockchain infrastructure. Recall that one of the core services in ASSURED is the trust establishment between entities, for cooperatively executing safety-critical functions, through the provision of assurance claims that can verify their Actual Level of Trustworthiness (based on the use of attestation enablers). Thus, the focus here is the management and sharing of all calculated attestation-related data, which is performed through the use of smart contract, while guaranteeing the compliance to the data confidentiality, data integrity and data traceability requirements and models defined in WP1.
Keywords	Data Confidentiality, Data integrity, Privacy, Trust, Attribute0based Encryption

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	Description of the core crypto primitives, considered in ASSURED, including symmetric and asymmetric crypto, hash algorithms, privacy-preserving signatures, etc. and their mapping to the core ASSUREd services (Chapter 2)	Kaitai Liang, Shihui Fu (TUDE) Nada El Kassem (SURREY) Stefanos Venios (S5)
v0.3	14.01.2022	First draft of the detailed Secure Device Registration and Enrollment protocol with all the interactions with the TPM-based Wallet (Chapter 4)	Nada El Kassem, Liqun Chen (SURREY) Edlira Dushku, Benjamin Larsen (DTU) Dimitris Papamartzivanos, Dimitris Karras (UBITECH)
v0.4	21.01.2022	First draft of the detailed ABE scheme sequence diagrams as well as the remote attestation diagrams (Chapter 4)	Kaitai Liang, Shihui Fu (TUDE) Nada El Kassem, Liqun Chen (SURREY)
v0.5	04.02.2022	First draft of the revocation protocol sequence diagrams as well as description of the high-level functionalities for all specified ASSURED secure data management protocols (Chapter 3 and 4)	Nada El Kassem, Liqun Chen (SURREY) Edlira Dushku, Benjamin Larsen (DTU) Dimitris Papamartzivanos, Dimitris Karras (UBITECH)
v0.6	11.02.2022	Update and finalization of all sequence diagrams including the execution of specific TPM commands (Chapter 4)	Nada El Kassem, Liqun Chen (SURREY) Edlira Dushku, Benjamin Larsen (DTU) Dimitris Papamartzivanos, Dimitris Karras (UBITECH) Sotiris Kousouris, Stefanos Venios (S5)
v0.7	18.02.2022	Description and mapping od the use of each one of the ASSURED secure data management services in the context of the use cases (Chapter 5)	Thanassis Giannetsos (UBITECH) Kaitai Liang, Shihui Fu (TUDE) Nada El Kassem (SURREY) Sotris Kousouris (S5)
v0.9	24.02.2022	Review the document	Liqun Chen (SURREY) Dimitris Papamartzivanos (UBITECH)
v1.0	27.02.2022	Finalisation of the document	Thanassis Giannetsos (UBITECH), Nada El Kassem (SURREY)

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)

Funded by the Horizon 2020
Framework Programme of the European Union

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.

COPYRIGHT NOTICE

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

Following the compilation of Deliverable D1.1 [17] which documented the **technical security, privacy and trustworthiness requirements** that need to be achieved by the ASSURED framework, when it comes to the **management of both operational and threat-intelligence (attestation) data**, D4.2 proceeds with the description of all **cryptographic primitives and protocols for safeguarding the secure communication (between devices) as well as the secure and privacy-preserving data sharing over the ASSURED Blockchain infrastructure**. Recall that one of the core services in ASSURED is the trust establishment between entities, for cooperatively executing safety-critical functions, through the provision of assurance claims that can verify their Actual Level of Trustworthiness (ATL). Such claims are enabled by the novel attestation schemes of ASSURED that not only enable such a dynamic trust assessment but also allow the monitoring of the correct configuration and behavioural state of each device in the target service graph chain. This latter feature is essentially based on data insights that can be extracted from the continuous attestation reports produced throughout the entire "*Systems-of-Systems*" ecosystem.

In this context, the **management and sharing of all calculated attestation-related data is performed through the use of smart contracts and DLTs**, as described in D4.1 [14]. *Recall that one of the core innovations of ASSURED is the employment of such decentralized infrastructures for the certifiable and auditable sharing of both attestation policies as well as attestation results and their accompanying system traces* based on which the verification was executed; embodying the monitored configuration and execution system properties depicting the correctness of a device (not compromised). Therefore, the creation of a digital "**attestation data hub**", based on the use of DLTs, is considered as a crucial building block in ASSURED.

However, considering the sensitive nature of this type of threat-intelligence data, not safeguarding their security and privacy might prove a challenging hurdle that can limit the applicability of such a solution. Consider, for instance, the implications in the case of an adversary that was able to get unauthorized access to the attestation evidence of a device that was used during its verification process: This could give her insights on the type of OS running in the device, the types of loaded binaries, and in general the device's execution profile which could lead to the identification of possible (vendor-specific) vulnerabilities. **What is needed is to build new on- and off-chain data management models and services of enhanced security, privacy and data protection and the technologies that encode them**.

This is the core focus of D4.2 which presents the concrete functionalities and algorithms, designed in ASSURED, for enhanced data security, integrity, and privacy-preserving mechanisms. This enables **advanced data privacy and ownership safeguarding** (**privacy by design**) and **data provenance and sovereignty checking mechanisms.** Understanding the functionalities and security requirements for the general technical framework enables us to narrow down the focus towards safeguarding the data flow and data sharing among the envisioned use cases (as defined in D1.4 [15]) through implementation of concrete cryptographic algorithms.

To this end, this deliverable presents the core cryptographic primitives, adopted in ASSURED, based on which all offered services exhibit the required guarantees on integrity, confidentiality, and authenticity of the exchanged data: From fleshing out the complete **Device Registration & Enrollment protocol**, that a device needs to first execute for correctly creating all of the appropriate cryptographic keys, to the **security and privacy protection of the exchanged operational and/or attestation-related data based on the type of sensitive information included**. These cryptographic primitives range from simple hash functions, conventional encryption and signature

schemes to more complex privacy preserving signatures such as Direct Anonymous Attestation (DAA) and group signatures.

For the former, ASSURED combines the aforementioned cryptographic ingredients in combination with the **TPM-based Wallet**, loaded in each device, towards designing a complete protocol that allows a **device to certify the validity of its underlying Trusted Platform Module (TPM) prior to creating all of the necessary cryptographic material, binded under strong key usage protection policies**, in order to get access to the attestation data hub (via the Blockchain Certification Authority) and be able to securely participate in the later secure data exchange and sharing. For the latter, ASSURED designs a novel **Attribute-based Encryption (ABE) scheme** for protecting the confidentiality of attestation-related information and release them to stakeholders with **different levels of access and information granularity**. This scheme is completely decentralized (bypassing the limitation of existing centralized ABE schemes) leveraging the capabilities of the TPM as a root-of-trust. Finally, in the case of a device misbehaviour, ASSURED has also prompted for the design of a **secure and privacy-preserving revocation protocol** for de-activating all of the device's credentials, thus, excluding it from any further participation in the system. All such schemes do not only enable the necessary on- and off-chain secure interactions for capturing the security and privacy data sharing requirements for the envisioned use cases (defined in D1.4 [15]) but are also efficient enough to offer real-time robustness and convenience for online supply chain operations

Beside the presentation of all the crypto details of such schemes, with their exact sequence diagrams, D4.2 also presents a concrete mapping between the high-level security requirements (as defined in D1.1 [17]) to these designed cryptographic protocols. As aforementioned, such high-level requirements actually depict the functional specifications of the ASSURED framework when it comes to the secure data management and, thus, D4.2 documents how these functionalities are enabled by the designed protocols. Based on this detailed mapping, D4.2 describes how the data sharing behaviours of all actors, in each one of the envisioned use cases, are facilitated through the advanced ASSURED secure data management protocols.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Scope and Purpose

As described in D4.1 [14], ASSURED Blockchain infrastructure provides the framework for the **secure (attestation) data management and information exchange**, as needed by the dynamic trust assessment services. In this context, what is crucial is to build new **on- and off-chain data management models and services of enhanced security, privacy and data protection** and the technologies that encode them. The endmost goal is to offer the necessary confidentiality, integrity and privacy-preserving guarantees for all ASSURED core services ((as were fleshed out in D1.1 [17]) and summarized in the following table) that rely on secure data exchange:

Security service	Description
Data confidentiality of operational data	It should be ensured that the data exchanged during communication between devices is carried out in a **secure and authentic** way. Thus, the necessary cryptographic material should be established for safeguarding any data communication: either *direct* or *indirect* that takes place through the ASSURED DLTs.
Data integrity and verification of the attestation data	The execution of attestation processes requires the **secure exchange of attestation evidence** between the Prover and the Verifier. This information will also be shared with the Security Context Broker (SCB) for **certifiable recording** on the edger (enabling the creation of an "**attestation data hub**") and **authorized and accountable sharing** to only allowed internal and/or external stakeholders [24].
Privacy protection of the exchanged operational and/or attestation data based on the type of sensitive information included	Depending on the operational needs, different levels of **privacy configuration** should be supported capturing the needs (e.g., full vs partial anonymity, unlinkability, untraceability, etc.) of all participating devices and users in the context of a specific service graph chain. As it pertains to the attestation-related data, **attestation evidence privacy** should be enabled: rovers should not be required to expose configuration and execution details as part of the measurements sent to the Verifier. Attestation and verification should be achieved in a privacy-preserving manner by enabling Provers to attest to their device configuration correctness and/or a critical program's correct execution in **zero-knowledge** while enabling the entire network to benefit from the security guarantees of the ASSURED remotely verifiable attestation enablers.
Data confidentiality of accompanying system/attestation raw data	The control-flow data or configuration traces that are used in their respective attestation algorithms should not be disclosed, in the interest of retaining the privacy of the Prover.
Release of attestation-related information, to stakeholders, with different levels of access and information granularity	ASSURED implements an Attribute-based Encryption (ABE) scheme, where it is possible to encrypt data depending on the attributes of the party that is intended to decrypt the data. Thus, it is possible to disclose data to different parties with different levels of granularity, based on the security and privacy requirements.

Secure design of a new device to be securely on-boarded to the overall environment	In order to securely enroll and register a device to the system, we need to verify the correctness of its state prior to its enrollment, as well as to be able to certify the correct design, meaning that the correct components are loaded, and the device ID and credentials are certified.

Table 1.1: ASSURED Secure Data Sharing & Management Service Requirements

Based on the detailed data sharing behaviours and models defined in D1.4 [15] (capturing the data sharing requirements of all actors in the envisioned use cases), this deliverable defines the necessary cryptographic protocols for: (i) Devices to be **securely on-boarded** in the target "*Systems-of-Systems*" environment so that the appropriate **cryptographic material** can be created (i.e., Attestation Key (AK), DAA Key, Pseudonyms, etc.), by the device TPM-based Wallet [23], as well as the **issuance and management of Verifiable Presentation (VPs)** based on credentials that include all the necessary and verified attributes for getting access to all of the offered services, (ii) Devices recording (in a **certifiable and auditable** manner) the attestation outputs - yielded from the execution of the attestation enablers - on the ledger so that the SCB can record the collected attestation reports into attestation history chains (per device) and protect them against unauthorized access so that only authenticated stakeholders can request reading privileges (i.e., other supply chain vendors, OEMs, certification authorities, etc.), and (iii) Devices should be able to encrypt such decentralized attestation data flows, in an efficient manner, so that further sharing can occur to only authorized entities that can exhibit the correct attributes.

In this context, D4.2 documents all of the crypto primitives leveraged for protecting the aforementioned data sharing behaviours and guarantee their compliance to the data confidentiality, data integrity and data traceability requirements of ASSURED (Table 1.1). For capturing all requirements regarding the management and sharing of attestation-related data, the ASSURED Security Context Broker (SCB) [24] offers **trusted ledger operations** and the deployment and enforcement of attestation policies through smart contracts [19]. Before a device can download and execute an attestation policy, it needs to have been registered to the overall system and the Blockchain Certification Authority (Blockchain CA) so as to get the appropriate certificate. This is done through the device's TPM-based Blockchain Wallet that executes the device registration and enrollment protocol first with the Privacy CA (for certifying the validity of the TPM) and the Blockchain CA. During the 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 can only be used only that device is in a correct (authorized) state (Section 3.2 and 4.1). This is based on the **policy-protected key usage attribute** described in D1.3 [13] as part of the overall ASSURED secure key management landscape (Section 3.1). In this end, Deliverable 4.2 presents a **secure (three-phase) enrollment protocol that explains the concrete issuing of the CA and the Blockchain credentials to ensure that a TPM wallet can only execute signing operation if the platform is configured correctly with an authorized tracer**.

Once the public part of a device's AK has been validated and published as part of a transaction on the ledger when executing an attestation policy and sign the attestation result (see [14] for more details), any other device can ascertain its correctness in an oblivious manner using a simple challenge-based remote attestation protocol [33], where the Verifier sends the Prover a fresh challenge (nonce), and if the Prover replies with a signature over the challenge using its secret AK, then the Verifier confirms that the Prover is in a correct state. In this context, in what follows, we will discuss and present the concrete cryptographic protocol that support the execution of such remote attestation protocols (be it either CFA or CIV [16]) supporting the TPM-Tracer internal communication for signing authorized traces.

Furthermore, ASSURED employs a hybrid approach where while the attestation results are securely stored and shared on the ledger (thus, creating the envisioned *attestation data hub*), the attestation evidence are not stored on the Blockchain for performance issues. Instead, ASSURED makes use of an efficient data storage engine [21]. Hashed attestation results are stored on the ledger whereas the encrypted system traces (control-flow traces or digest lists of loaded binaries) are securely stored in the ASSURED Data Storage Engine and an appropriate pointer is created and ammended to the attestation record.

Regarding the private ledger, the core information kept is the **hash of an attestation operation** that has been performed at the device level, and a pointer to the encrypted off-chain attestation data object, allowing the ledger to be more lightweight. The attestation hash is provided by the devices as a result of execution of specific attestation contracts, while the pointer to the off-chain storage is provided by the SCB that handles the operations of storing the encrypted attestation data to the database. ASSURED will also deploy a Searchable Encryption Component presented in D4.3 [21] that will allow interested stakeholders to perform queries on top of encrypted metadata that are stored in the public ledger and accompany the different attestation information generated by the devices. These metadata will be provided by the SCB in an encrypted manner.

After querying the encrypted attestation evidence, the decryption is achieved at the device/user level and is possible only if the set of attributes of the user key matches the attributes of the ciphertext. User attributes are been issued during the secure device registration phase, by the Privacy Certification Authority (Privacy CA), which then credits the device a credential that can forward to the SCB for verification. If successful, the SCB will notify the Blockchain Certification Authority (Blockchain CA) which, in turn, will create the appropriate certificate for the device including all verified attributes based on which it can access specific attestation result. This is then forwarded to the Membership Service Provider (MSP) that verifies that all required attributes are included in the user's certificate.

After a Blockchain user being authenticated to a ledger, it can only have decryption rights to the encrypted data stored on the ledger only if the user possesses some attributes that make correct decryption. **This allows data owners to share data safely with the designated group of users rather than a third party or other users**. In the ASSURED framework, the TPM-based Wallet is used to support a **Key-Policy Attribute-based Encryption (KP-ABE) where the secret key of a Block chain user and the ciphertext are dependent on the user attributes**.

Finally, ff any of the edge devices are compromised, then ASSURED framework will offer **secure and efficient revocation of all device's credentials** so that its further participation to the system is prohibited. Revocation is a standard consideration for any ICT system and supply chain. In case of a misbehaving platform, the wrongdoer can be evicted and be prevented from performing any further damage to the system. Supporting this requirement, Deliverable 4.2 will discuss a concrete revocation scheme that enables an edge device to deactivate any credentials and short-term signature keys that were created during their enrollment phase in the ASSURED ecosystem.

1.2 Relation to other WPs and Deliverables

In what follows, Figure depicts the relationship of the deliverable with other Work Packages (WPs) as well as the other tasks in the same WP(4). As aforementioned, the main purpose of this doc-

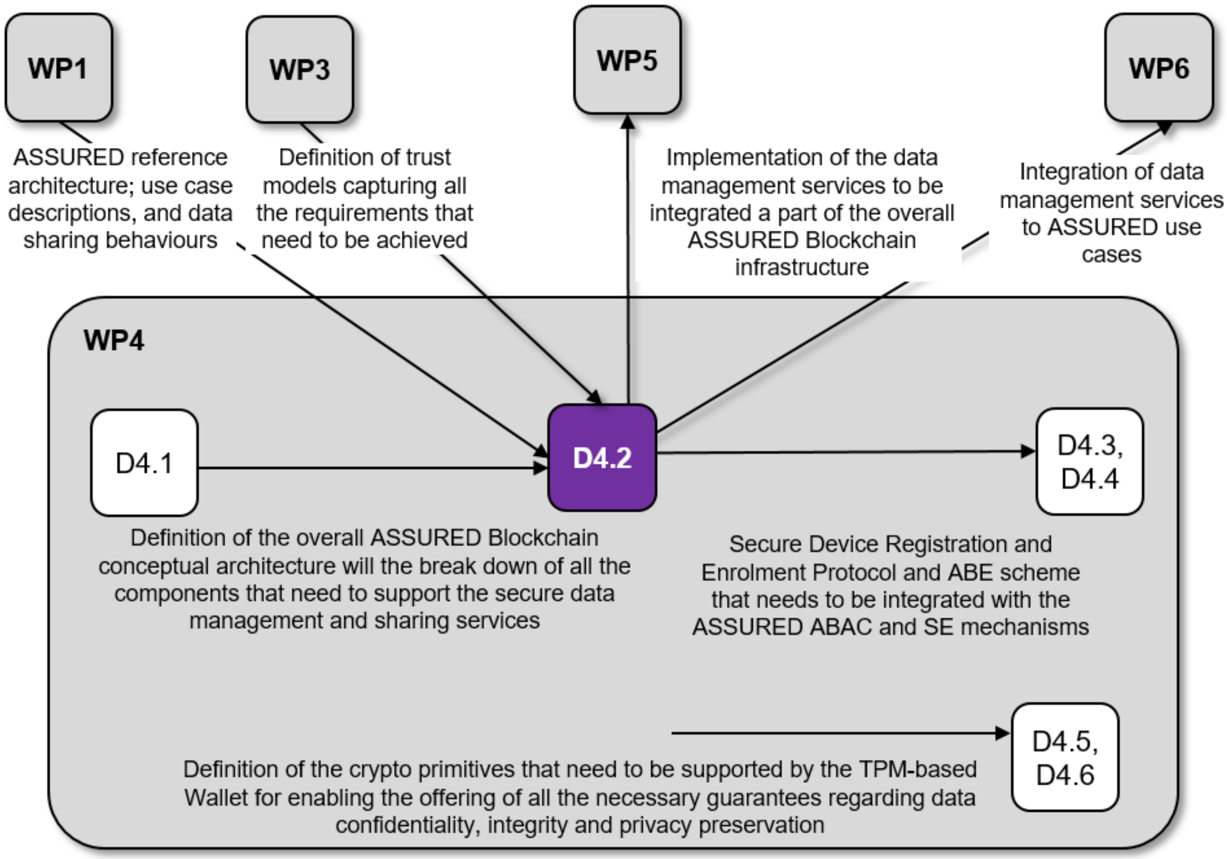

WP1
WP3
WP5
WP6
ASSURED reference architecture; use case descriptions, and data sharing behaviours
Definition of trust models capturing all the requirements that need to be achieved
Implementation of the data management services to be integrated a part of the overall ASSURED Blockchain infrastructure
Integration of data management services to ASSURED use cases
WP4
D4.1
D4.2
D4.3, D4.4
Definition of the overall ASSURED Blockchain conceptual architecture will the break down of all the components that need to support the secure data management and sharing services
Secure Device Registration and Enrolment Protocol and ABE scheme that needs to be integrated with the ASSURED ABAC and SE mechanisms
D4.5, D4.6
Definition of the crypto primitives that need to be supported by the TPM-based Wallet for enabling the offering of all the necessary guarantees regarding data confidentiality, integrity and privacy preservation

Figure 1.1: Relation of D4.2 with other WPs and Deliverables

ument is put forth all of the details of the crypto protocols designed in ASSURED for supporting the secure data management and sharing services. Thus, in this context, all defined schemes need to be aligned with the trust models defined in D1.3 [18] that capture all of the relationships between all actors in the target ecosystems as well as the specific confidentiality, integrity ans privacy-preserving requirements. Furthermore, the ASSURED on- and off-chain secure data management needs to be able to support all of the data sharing behaviours defined in D1.4 [15] for the envisioned use cases. This includes both operational and attestation-related data that, depending on the safety-critical nature of each service. might entail different level of security and privacy to be achieved. Thus, D4.2 needs to provide all advanced crypto primitives 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.

Furthermore, in the context of WP3, all designed attestation enablers (as defined in D3.2 [16] and D3.6 [22]) need to employ these assured secure data management services. Thus, D4.2 also needs to investigate how this integration can be done and offer the necessary secure communication protocols.

The outcome of Deliverable D4.2 is also intended to support the definition of later activities in WP4. As aforementioned, all secure data management operations are supported by a device's TPM-based Wallet which is defined in D4.5 [23], thus, the appropriate definition of the type of cryptographic material needed should be provided. In the same direction, the integration of such crypto protocols with the Attribute-based Access Control (ABAC) and Searchable Encryption (SE) schemes (defined in D4.3 [21]) should be documented.

1.3 Deliverable Structure

This deliverable is structured as follows: **Chapter 2** describes the cryptographic building blocks consisting of hash functions, encryption schemes, digital signatures and privacy preserving signatures. These will constitute the main blocks for building ASSURED cryptographic protocols which guarantee the security and safety requirements related to data integrity, confidentiality, and authenticity so as to have verifiable evidence on the correctness of the produced and transmitted data between ASSURED Blockchain users. **Chapter 3** provides a description of ASSURED newly designed protocols needed for the DLT Data Management. This chapter describes ASSURED secure device enrollment protocol with the processes of issuing the device's credentials and authenticating new joining devices. It also describes ASSURED Key-Policy Attribute-based Encryption (KP-ABE) and Control Flow Attestation (CFA) protocols that will be supported by the TPM wallet. Finally a novel revocation mechanism for revoking devices in a privacy preserving manner will be presented in this chapter. **Chapter 4** presents sequence of diagrams that explain the exact cryptographic flow of actions between ASSURED entities for each of the protocols described in Chapter 3. **Chapter 5** provides a detailed mapping of how ASSURED protocols that were described in Chapters 3 and 4 meet the secure data sharing requirements. Based on this detailed mapping, Chapter 5 then describes how the data sharing behaviours of all actors, in each one of the envisioned use cases, are facilitated through the advanced ASSURED secure data management protocols, as defined in Chapters 3 and 4. Finally, **Chapter 6** concludes the deliverable and suggests any future work.

Chapter 2

Crypto Primitives

The main goal of ASSURED is to design cryptographic protocols that ensure the security and safety requirements related to data integrity, confidentiality, and authenticity so as to have verifiable evidence on the correctness of the produced and transmitted data between ASSURED block chain users. The data should only be shared through encryption protocols so as to safeguard them from communication attacks. ASSURED framework will help in creating this safe communication channel between authenticated parties, attesting each device involved in the exchange. To further enhance the transmission security, the actions performed during the exchanged should be registered on a distributed ledger.

ASSURED crypto protocols will be the building blocks for a secure attestation data exchange engine via a Blockchain ledger and the trusted Blockchain wallet. The main challenge of such cryptographic designs is to provide the required usability, efficiency, and security for ASSURED use cases. This will be achieved in ASSURED by developing effective crypto protocols that secure the data management through the Blockchain, smart contracts and trusted hardware functionalities. ASSURED cryptographic protocols will be efficient enough to offer real-time robustness and convenience for online supply chain operations for ASSURED use cases. This Chapter describes the basic cryptographic primitives that will be required by ASSURED to ensure data integrity, confidentiality, authenticity and privacy if needed.

2.1 Symmetric and Asymmetric Cryptography

Asymmetric cryptography is also known as public key cryptography. Asymmetric cryptography works with a pair of keys that is used for encryption/decryption or signing/verification purposes. The key pair consists of a public key with its corresponding private key. The public key is accessible by anyone, while the private key must be kept a secret from everyone but the creator of the key. This is because encryption and verification occur with the public key, while decryption and signing occur with the private key. This ensures that only the recipient can decrypt or sign the data, with their own private key. Symmetric encryption is a type of encryption where only one key, called a secret symmetric-encryption key, used to both encrypt and decrypt data. This encryption method differs from asymmetric encryption where a pair of different keys, one public and one private, is used to encrypt and decrypt messages.

ASSURED will investigate using symmetric or/and asymmetric cryptography depending on the security, efficiency and usability of each proposed ASSURED protocol. For instance, a symmetric-key algorithm that will be considered in ASSURED framework is the Advanced Encryption Standard (AES). For ASSURED Attribute Based Encryption(ABE) protocol, a combination of symmetric and asymmetric encryption schemes will be considered in order to achieve a high level of efficiency.

ASSURED uses asymmetric signing algorithms for attestation, enrollment and revocation protocols. An asymmetric algorithm identifier will indicate a family of algorithms and methods that are used with that algorithm. For an asymmetric algorithm, the methods of signing are dependent on the algorithm (RSA or ECC). For symmetric signatures, only the HMAC signing scheme is currently used in ASSURED authorization,authentication and ASSURED ABE protocols for integrity checks. If a key may be used for signing, then it will have an attribute to allow it for.

2.2 Encryption

There are many symmetric algorithms that use the same key for both encryption and decryption. An example of symmetric-key algorithm that may be considered in ASSURED framework is the Advanced Encryption Standard (AES). The TPM uses symmetric encryption to encrypt some command parameters (e.g.: authentication information) and to encrypt objects stored outside of it. Cipher Feedback mode (CFB) is the only block cipher mode required by TPM 2.0 Specification to encrypt command parameters (as well as sessions and sensitive area of a key object). Weak keys are not permitted to be used (some algorithms have known weak keys, if such a key is generated, it must be discarded, and a new key generated by starting over with another iteration).

Symmetric-key encryption is used for keeping data secret from all attackers, encrypting communications between different entities to secure communication channels. Since the TPM is a cryptographic co-processor which knows how to encrypt data using its secret symmetric keys, then these encryption can be done using the TPM embedded in ASSURED entities. Asymmetric encryption is mostly used in day-to-day communication channels, especially over the Internet. Popular asymmetric key encryption algorithm includes ElGamal, RSA, DSA, Elliptic curve techniques, PKCS.

A TPM uses asymmetric algorithms for attestation, identification, and secret sharing. An asymmetric algorithm identifier will indicate a family of algorithms and methods that are used with that algorithm. The only supported asymmetric algorithms in TPM 2.0 are RSA and Elliptic-Curve Cryptography (ECC) using prime curves. A TPM is required to implement at least one asymmetric algorithm.

Symmetric and asymmetric encryption are each better used for different situations. Symmetric encryption, with its use of a single key, is better used for data-at-rest. Data stored in databases needs to be encrypted to ensure it is not compromised or stolen. This data does not require two keys, just the one provided by symmetric encryption, as it only needs to be safe until it needs to be accessed in the future. Asymmetric encryption, on the other hand, should be used on data sent to other people. If only symmetric encryption were used on data such as in email, the attacker could take the key used for encryption and decryption and steal or compromise the data. With asymmetric encryption, the sender and recipient ensure only the recipient of the data can decrypt the data, because their public key was used to encrypt the data. Both types of encryption are used with other processes, like digital signing or compression, to provide even more security to the data.

2.3 Hash Functions (HMAC)

A hash function is a unique identifier for any given piece of content. It's also a process that takes plaintext data of any size and converts it into a unique ciphertext of a specific length. The output generated is called hash value, hash or digest. The main characteristics of a cryptographic hash function are that given a message, it is easy to compute the hash; given the hash, it is difficult to compute the message; and that given a message, it is difficult to find a different message that would produce the same hash (this is known as a collision).

Hash functions in ASSURED are used for data integrity and often in combination with digital signatures. Hash functions may be used in ASSURED framework directly by external software or as the side effect of many TPM operations. The TPM uses hashing to provide integrity checking and authentication as well as one-way functions, as needed (such as, KDF).

In order to guarantee that only trusted and compromised devices can participate in envisioned supply chain ecosystem, all involved devices will use the TPM secure boot mechanism and their trust level will be continuously attested and assessed. To this end, all signatures on attestation data will include 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, which will allow any other party to check whether the data stems or was acknowledged by a trusted entity. Even a 1-bit change in the PCR input for the hash will produce completely a different hash. Any change with the original PCR value, will lead to different hash function output. The attestation will fail and show that the PCR has been modified.

Informally, a hash function is an efficiently computable function whose description is fully public. There are no secret keys and anyone can evaluate the function. Let H be a hash function from some large message space $\mathcal{M}$ into a small digest space $\mathcal{T}$. We say that two messages $m_0, m_1 \in \mathcal{M}$ are a collision for the function H if

$$H(m_0) = H(m_1) \text{ and } m_0 \neq m_1.$$

We say that the function H is collision resistant if finding a collision for H is difficult. Since the digest space $\mathcal{T}$ is much smaller than $\mathcal{M}$, we know that many such collisions exist. Nevertheless, if H is collision resistant, actually finding a pair m_0, m_1 that collide should be difficult.

The Secure Hash Algorithm (SHA) standard describes a number of hash functions that offer varying degrees of collision resistance. For example, SHA-256 is a function that hashes long messages into 256-bit digests. It is believed that finding collisions for SHA-256 is difficult.

An important application for collision-resistant hash function is its ability to extend primitives built for short inputs to primitives for much longer inputs, such as Hash based Message Authentication Code (HMAC). HMAC is a type of a message authentication code (MAC) that is acquired by executing a collision-resistant hash function on the data (that is) to be authenticated and a secret shared key. Like any of the MAC, it is used for both data integrity and authentication. Checking data integrity is necessary for the parties involved in communication. HTTPS, SFTP, FTPS, and other transfer protocols use HMAC. The cryptographic hash function may be MD-5, SHA-1, or SHA-256.

2.4 Digital Signatures

A digital signature is a cryptographic protocol used to verify the authenticity and integrity of data in ASSURED by ensuring the trustworthiness and accuracy of data exchange among diverse systems that support ASSURED Block chain technology. ASSURED block chain users should be convinced that the data they have received or provided was not modified by an unauthorized party. Data integrity will be achieved by means of cryptographic digital signatures for creating attestation reports that will uploaded on ASSURED ledger and verified by block chain parties.

ASSURED investigates the adoption of key technologies, in the fields of trusted computing and lightweight cryptographic trust anchors, as enablers for the secure configuration, deployment, operation and verifiable computing of critical programmable components running at the edge and the secure communication and data sharing amongst them and with other interested (and authenticated) stakeholders acting as data seekers. For instance, an ASSURED tracer that measures the traces of an edge device should be able to prove that these supplied traces originate from an authentic tracer. An edge device, using its embedded TPM, should be able to verify the authenticity of the tracer. This can be achieved by letting the ASSURED tracer to create a digital signature on the traces, this signature should be verified by the TPM before creating an attestation report on the device state.

The TPM produces a message digest of the traces then uses its restricted key to sign this generated message. A key may be restricted to sign messages with specific contents. When a key has this restriction, the TPM will not use the key to sign message digests that the TPM did not compute or any digest that is provided by a non-authorized tracer in case of producing control flow attestations.

A signing scheme can be used when a key allows for it because not all schemes are valid for all keys. A TPM generates an error if the scheme is not allowed with the indicated key type. A restricted signing key requires to have a signing scheme specified in the key definition and that is the only signing scheme that is allowed to be used with the key. Unrestricted keys may contain a signing scheme selection, or the signing scheme may be determined when the key is used. Attestations (signatures on the digests) are produced by the TPM to identify that the data as being signed by a legitimate TPM. If a restricted TPM key is used to sign this data, then any relying party can have assurance that the message data and the signature came from a legitimate TPM.

Digital signatures provide data integrity which is the main security ingredient for ASSURED use cases. For instance, in the context of the digital security of smart satellites use case the operation of a CubeSat constellation and especially the execution of the mission application is of high critically for the secure execution of the whole mission. A cooperation of devices and services is required, and it is important to ensure the integrity of all these parts (H/W S/W) in order to achieve secure execution of the mission application. The remote software update of the CubeSat is one the most critical parts for the operation of the CubeSats and thus for the whole digital security of smart satelites use case. This is a task performed by the CubeSat operator and it involves the upload of the updated mission S/W from the GroundStation to CubeSat through the file transfer service. Next we will introduce the cryptographic concept of the digital signature and its security requirements.

A digital signature algorithm allows for two distinct operations:a signing operation, which uses a signing key to produce a signature over raw data and a verification operation, where the signature can be validated by a party who can verify the signature with the knowledge of the public key that corresponds to the signing key used to generate the signature. Digital signatures can be built from symmetric or asymmetric cryptography. An asymmetric key consists of a public/private key pair. The private key is used to create a signature, and the corresponding public key is used to verify the signature, whereas the symmetric key is a shared secret key between the signer and the Verifier.

For example, in ASSURED a digital signature is required whenever binaries are loaded in an ASSURED device that digitally signs the binaries using the device's private key. Any remote Verifier can check its validity by using the public key corresponding to the private key. If the binary's signature is not valid, the binary has been tampered with and/or corrupted. Another example is validating the subject of a certificate issued by a Certificate Authority (CA). A CA issues a certificate to a subject based on the subject's ownership of the private key portion of a public/private key. The certificate contains a digital signature created with the subject's private key. The certificate also contains the subject's public key portion of the public/private key. If the signature does not verify under the users public key, or if the verification rules prescribed by the certificate are violated, the signature will be found invalid.

A digital signature scheme (DSS) is a tuple (KeyGen; Sign ; Verify) where KeyGen(n) outputs the secret-key sk and public-key pk, Sign (m) takes as input a message m and outputs a corresponding signature σ using the user secret key sk. Verifypk (m, σ) takes as input the message m and signature σ and outputs 1 if and only if $(m; \sigma)$ is a valid message/signature pair, otherwise outputs 0.

2.4.1 EU-CMA Security for a Digital Signature

We recall the definition of the EU-CMA (Existential Unforgeability under a Chosen Message Attack) security of a digital signature scheme [30]. The security of a signature scheme DSS=(KeyGen; Sign(sk) ; Verifypk, σ, m) is defined through game which is run between a simulator S and an adversary $\mathcal{A}$. In this game a pair of signing and verification keys (sk, pk) is generated by running the key generation algorithm KeyGen. Then, adversary $\mathcal{A}$ is given the verification key pk and provided with access to a signing oracle $sig_{sk}(\cdot)$. For each message m that $\mathcal{A}$ sends to the oracle via S, the oracle responds with a signature $\sigma = sig_{sk}(m)$. Eventually, $\mathcal{A}$ terminates its execution and outputs a message and signature pair (m^*, σ^*). The experiment returns 1 if σ^* is a valid signature on m^* under pk, i.e., Verify$_{pk}(\sigma^*, m^*) = 1$, and the message m^* was never queried to the signing oracle. The experiment returns 0 otherwise.

2.5 Privacy Preserving Signatures

ASSURED attestation process should allow for verifying the correctness and trustworthiness of a "*Systems-of-Systems*" rather than individual systems. This is achieved by a newly developed remote attestation variant called swarm attestation (check D3.6 [22] for more details). This means that attestation protocols, which run between individual entities, must be used as a building block to be integrated into an overall system. This requires, for instance, hierarchical privacy preserving attestation where one attested component/entity acts as Verifier for another. In this section we will describe different types of privacy preserving signatures that may be used as building blocks for establishing ASSURED swarm attestation.

2.5.1 Group Signatures

Group signature aims to provide a way to guarantee that a message was sent by a group member (authentication and data integrity) without leaking any information about which group member signed the message (privacy) unless an opening authority decides to open the signature as described in [10], [3] and [9]. More precisely, group signatures allow members of a group, which is administered by a group manager, to anonymously sign messages on behalf of that group. The signature Verifier will be convinced that the signature comes from some group member, but without knowing the signer. At the same time, an authority is able to determine the signer's identity (traceability) using some trapdoor information known as the signature opening operation.

Security properties of Group Signatures: The basic requirements of a group signature scheme is that any honest signature generated by a group member should be accepted as correct, also the signature should be traceable to the group member who issued it. Group signatures are also required to be anonymous, meaning that without the tracing key, it should be infeasible for an adversary (even given all the signing keys) to determine the identity of the group member who issued a specific signature. A second important requirement for group signature scheme is that it should be difficult for an adversary who can corrupt some set of group members C to output a valid signature that can't be traced to some member in C, this means traceability.

Syntax of Group Signature Schemes

A group signature scheme *GS=(G.Key Gen, G.Sign, G.Vrfy, G.Open)* consists of four polynomial-time algorithms:

1. The randomized group key generation algorithm *G.Key Gen*$(1^n, 1^N)$ takes inputs $1^n, 1^N$, where $n \in \mathbb{N}$ is the security parameter and $N \in \mathbb{N}$ is the group size. This algorithm returns (PK, TK, gsk), where PK is the group public key, TK is the group manager's tracing key, and gsk is a vector of N signing keys such that $gsk[i]$ corresponds to the i^{th} user signing key.
2. The group signature algorithm *G.sign*$(gsk[i], m)$ is a randomized algorithm that takes the i^{th} user secret key $gsk[i]$ and a message M, outputs a signature σ on m.
3. The group signature verification algorithm *G.vrfy*(PK, m, σ) is a deterministic algorithm that takes the group public key PK, a message m and a signature σ on m, returns 0 or 1.
4. The opening algorithm *G.Open*(m, TK, σ) is a deterministic algorithm that takes a message m, the tracing key TK and a signature σ as inputs, and returns an identity $i \in [N]$.

The general security notions for group signatures based on [4] are:

- **Correctness**: Correctness guarantees that an honest user can enroll in the group and produce signatures that are accepted by the Verify algorithm.
- **Anonymity**: Group signatures should be anonymous and don't reveal the identity of the group member who produced them. Since the group manager has the ability to trace signers, we must assume the group manager to be honest for anonymity to hold, but some of the other users may be malicious.
- **Traceability**: Traceability in group signatures was purely with respect to identifying a signer but did not require a proof of correct opening. Bellare et al. [3] included the use of a proof for correct opening that can be verified by anybody using a Judge algorithm. Eliminating the proof of correct opening can be done in case of putting trust on the group manager.
- **Opening Soundness**: This property guarantees that the opening of any group member signature's should not be attributed to somebody else.
- **Opening Binding**: This property guarantees that even if all authorities and users collude they should not be able to produce a valid signature that can be selectively traced back to different members.

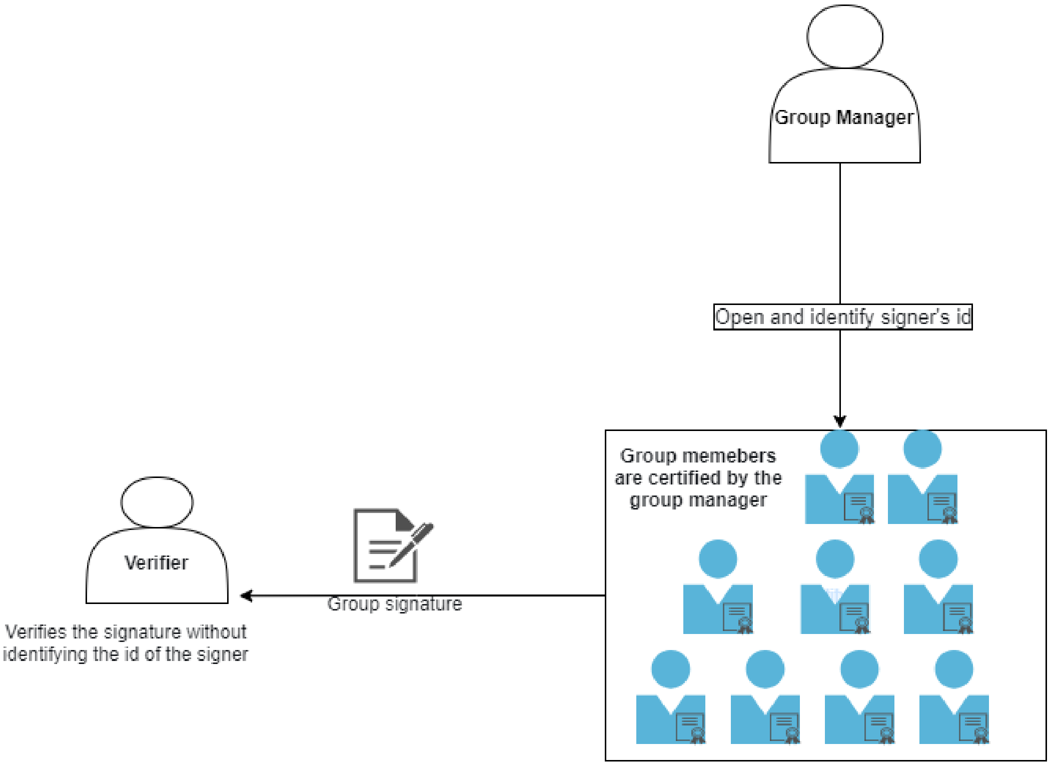

Group Manager
Open and identify signer's id
Group memebers are certified by the group manager
Verifier
Group signature
Verifies the signature without identifying the id of the signer

Figure 2.1: Overview of the Group Signature

2.5.2 Ring Signatures

The concept of ring signatures was first introduced in 2001 Rivest, Shamir and Tauman [40]. Ring signatures are a type of anonymous digital signatures, where the signer takes a number of public keys, called the ring, and a secret key which corresponds to one of the public keys. When the signer outputs a signature, a Verifier can be convinced that a secret key corresponding to one of the public keys has been used for the signature, yet the Verifier cannot tell which secret key is used. While group signature requires an opening authority, ring signature enables a signer to generate a message anonymously in the name of others without a manager. The Verifier checks only the validity of the signature, but can not know the identity of the real signer. Ring signature has many applications, such as e-voting, e-money, and whistle blowing. Several efficient lattice-based ring signatures have been naturally constructed such as [1, 28], but all of them have large verification key sizes. In general, ring signatures have two main properties:

- Unforgeability: It is not possible to sign on behalf of a ring without knowing one of the associated secret keys.
- Anonymity: It is not possible to know the identity of the user that output a signature.

Syntax of Ring Signatures

The ring signature scheme can be described by the following triple of algorithms:

1. R.ParGen(1^n): A probabilistic polynomial time algorithm that takes a security parameter n, outputs a set of public parameters pp.
2. R.KeyGen (pp): A probabilistic polynomial time algorithm that takes a set of public parameters pp, and outputs a public key pk with the corresponding secret key sk.

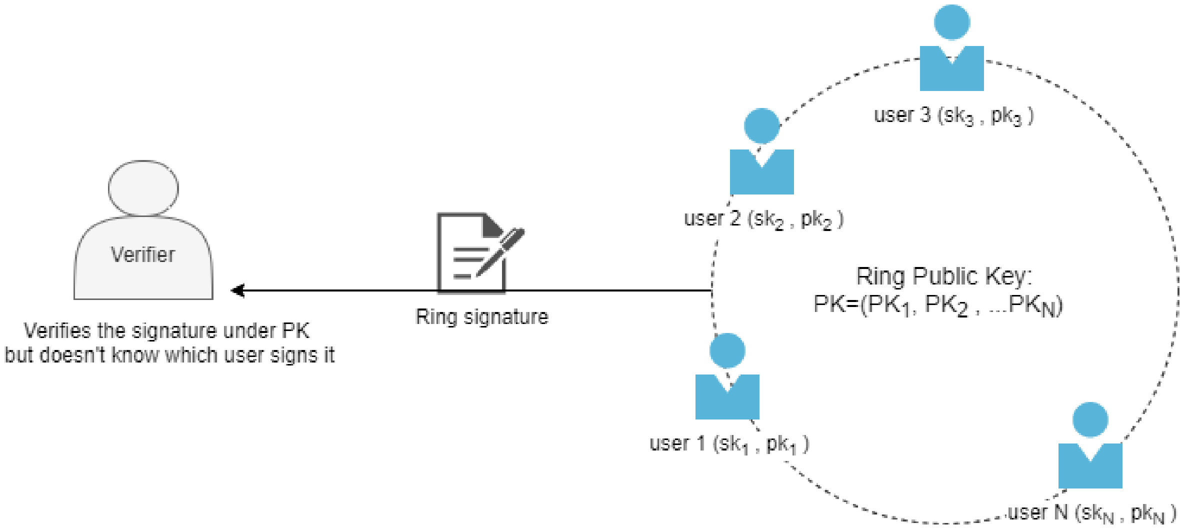

Verifier
Verifies the signature under PK
but doesn't know which user signs it
Ring signature
user 3 (sk$_3$, pk$_3$)
user 2 (sk$_2$, pk$_2$)
Ring Public Key:
PK=(PK$_1$, PK$_2$, ...PK$_N$)
user 1 (sk$_1$, pk$_1$)
user N (sk$_N$, pk$_N$)

Figure 2.2: Overview of the Ring Signature

3. R.sign(pp, sk, m, R): A probabilistic polynomial time algorithm that takes a set of public parameters pp, a signing key sk, a message m, and a set of public keys R, then returns a signature σ on m under SK.

4. R.verify$(\mathsf{pp}, \sigma, m, R)$: A deterministic algorithm that takes a set of parameters pp, a set of public keys R, a message m and a signature σ on m, outputs 1 (accept) or 0 (reject).

2.5.3 Direct Anonymous Attestation (DAA)

In general, a Direct Anonymous Attestation (DAA) scheme consists of an issuer(Privacy CA), a set of signers and a set of Verifiers. The issuer creates a DAA membership credential for each signer.A DAA credential corresponds to a signature of the signer's identity produced by the DAA issuer. A DAA signer consists of the (Host, TPM) pair. Their membership to the DAA community and trustworthy state is proved by providing the Verifier with a DAA signature of the TPM representation of the Host state. The DAA signature includes a zero-knowledge proof-of-knowledge, which is a cryptographic construct used to convince the Verifier that the signer possesses a valid membership credential, but without the Verifier learning anything else about the identity of the signer. In contrast to other privacy-preserving constructs, like group signatures [35,37], DAA does not support the property of traceability, wherein a group manager can identify the signer from a given group signature. In particular, the DAA is a protocol enabling a TPM and a Host device not only to authenticate themselves to a verifying edge device and to prove that the Host is in a trustworthy state, but also to do so in a privacy-preserving manner. More concretely, the DAA provides the TPM with the ability to sign its register values in an anonymous way, whilst still convincing the Verifier that it possesses valid DAA credentials.

Furthermore, when the DAA issuer also plays the role of a Verifier, the issuer does not obtain more information from a given signature than any arbitrary Verifier. However, to prevent a malicious signer from abusing anonymity, DAA provides two alternative properties as the replacement of traceability. One is the rogue signer detection, i.e. with a signer's private key, anyone can check whether a given DAA signature was created under this key or not. The other is the user-controlled linkability: two DAA signatures created by the same signer may or may not be linked from a Verifier's point of view. The linkability of DAA signatures is controlled by an input parameter called the basename. If a signer uses the same basename in two signatures, they are linked; otherwise, they are not.

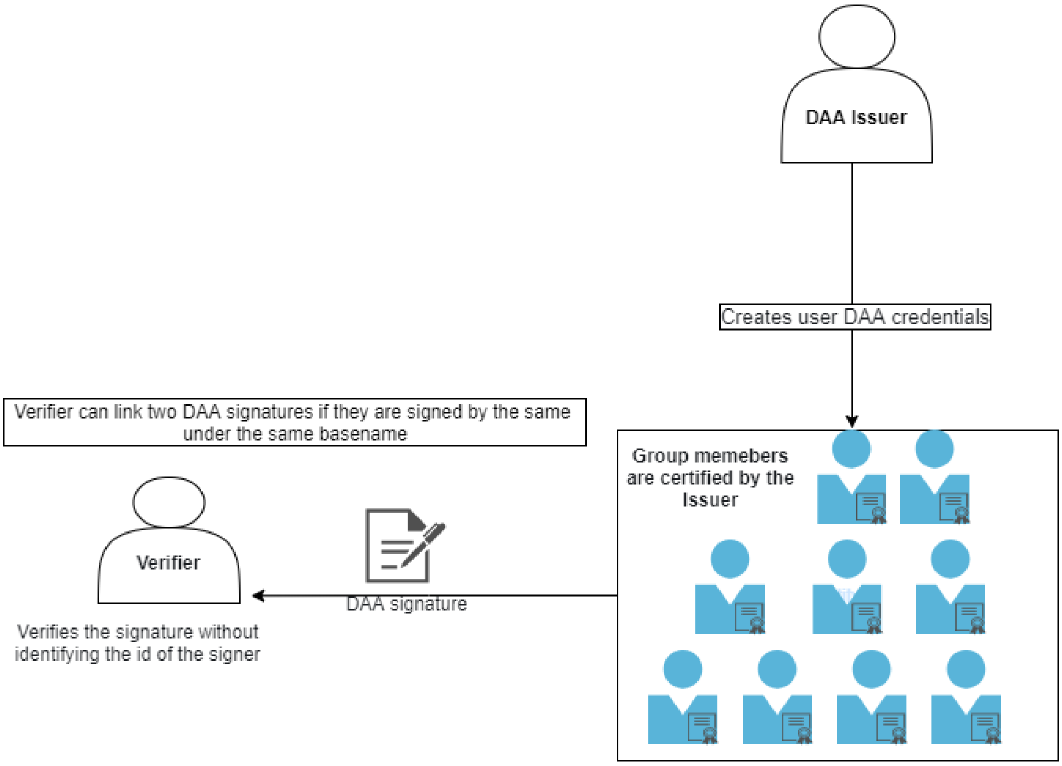

DAA Issuer
Creates user DAA credentials
Verifier can link two DAA signatures if they are signed by the same under the same basename
Group memebers are certified by the Issuer
Verifier
DAA signature
Verifies the signature without identifying the id of the signer

Figure 2.3: Overview of the DAA Signature

2.5.3.1 Security Properties of DAA

We summarize the following high-level security properties of DAA:

Unforgeability: When the Privacy CA and all TPMs are honest, no adversary can create a signature on a message μ with respect to basename bsn when no TPM signed μ with respect to bsn.

One-More-Unforgeability: When the Privacy CA is honest, an adversary can only sign in the name of corrupt TPMs. More precisely, if n TPMs are corrupt, the adversary can create at most n unlinkable signatures for the same basename.

Anonymity: An adversary that is given two signatures, with respect to two different basenames, cannot distinguish whether both signatures were created by one honest device, or whether two different honest devices created the signatures.

Non-frameability: No adversary can create signatures on a message μ with respect to basename bsn that links to a signature created by an honest device for the same basename, when this honest device never signed μ with respect to bsn.

The anonymity and non-frameability properties must hold even when the Privacy CA is corrupt. The existing DAA schemes implemented in the TPMs are based on either the factorization problem in the RSA setting or the discrete logarithm problem in the Elliptic-Curve (EC) setting. The concept of a DAA scheme was first proposed in 2004 by Brickell, Camenisch, and Chen [5]. This scheme is called RSA-DAA and supported by the TPM version 1.2. Later, Brickell, Chen, and Li proposed the first EC-DAA scheme based on symmetric pairings [6, 7]. Two EC-DAA schemes, based on asymmetric (Type 3) pairings, are supported by the TPM version 2.0 [8, 12].

2.6 Attribute-Based Encryption

ASSURED will adopt Attribute Based Encryption mechanism that involve techniques of issuing keys that bound to the devices' attributes. These keys are called attribute keys. Attribute keys consists of public (encryption keys) and private (decryption keys) portions.

In ASSURED framework, for maintaining adequate privacy, some secret information should only be shared within a subgroup of supply chain stakeholders. For instance, consider the case of the envisioned Public Safety use case where some extracted information and knowledge should be accessible by only a specific subset of municipality official bodies. Towards this direction, ASSURED should provide a desirable support for Attribute-based Encryption (ABE). In this case, any requested entity will be able to have access to the (encrypted) data but only those that can produce the necessary attributes should be able to decrypt the encrypted data.

Attribute-based encryption (ABE), introduced by Sahai and Waters [41], offers an expressive way to define asymmetric-key encryption schemes for policy enforcement based on attributes. Here both a user secret key and ciphertext are associated with sets of attributes. There are two flavours of ABE defined, i.e. ciphertext-policy attribute-based encryption (CP-ABE) and key-policy attribute-based encryption (KP-ABE). In CP-ABE, a user encrypts the data according to a predicate (access policy) defined over attributes, such that only the party that possesses a secret key associated with the attribute set satisfying the predicate is able to decrypt the ciphertext. In KP-ABE, the idea is reversed. Here the ciphertext is associated with the attribute set and the secret key is associated with the predicate defined over the attributes.

In practice, the integration of the access structure in either the keys or the ciphertexts (depending on which type of ABE we are dealing with) can be done by exploiting primitives such as secret sharing schemes, which allows us to share a secret value among different attributes. The idea is that the secret can only be recovered if the set of attributes satisfies the access policy. For instance, if our access policy consists of a set of n attributes, for which holds that we need at least $t \in [1, n]$ of those in order to satisfy the policy, then this can clearly be realized by applying Shamir's secret sharing scheme (SSSS), which is a (t, n)-threshold scheme. The first ABE scheme that was proposed in [41] applies this strategy.

Typically, in KP-ABE, the secret to be shared is some secret value that relates to the master secret key, which is usually some value α in the exponent of one of the public keys that is used to hide the message M in the encryption. This secret value is shared with SSSS, and the shares are each put in the exponent of some cleverly chosen generators. Then, the ciphertexts are related to a set of attributes in some fashion, for which holds that if there is a subset of attributes that consists of at least t attributes that match the set of attributes associated with the secret keys, then the secret keys can be used to decrypt the ciphertext. This works because the ciphertext also contains a secret value s, specified by the encryptor. In [41], Sahai and Waters define unique generators for each attribute in the system, such that this specific secret value is only 'added' to the ciphertext for those attributes that the encryptor wishes to use. This can be done by only raising the corresponding generators to the power of the secret value s. In CP-ABE, we can do something similar, but instead use SSSS in the sharing of the secret value in the ciphertext, whereas the keys use the unique generators raised to the power of the master secret key value α.

2.7 Classical and Post-Quantum Cryptography

The existing cryptographic schemes adopted in ASSURED are based on either the factorisation problem in the RSA setting or the discrete logarithm problem in the Elliptic-Curve (EC) setting. Since the factorisation problem and discrete logarithm problem are known to be vulnerable to quantum computer attacks [42], then all the existing standardized protocols may not be post quantum secure, i.e., an adversary with a quantum computer can subvert the existing protocols and might break their security. Given the rapid development of technology nowadays, powerful quantum computers might become a reality shortly soon. This has motivated the era of post-quantum cryptography (PQC), which refers to the construction of cryptographic algorithms to withstand quantum adversaries. Amongst many important areas in post quantum cryptography such as multivariate, symmetric, code-based, hash-based or lattice-based cryptography, lattice-based cryptography is significantly the most promising. Figure 2.4 explains the current *Quantum Safe* status of different cryptography, green coloured boxes refer to Quantum safe cryptography while yellow coloured boxes indicate that these types of cryptography needs some adjustments (indicated in the gray boxes) such as increasing key sizes or hash function output size to become Quantum Safe. Finally, red coloured boxes indicate the types of cryptography that are not Quantum Safe.

The US-based National Institute of Standards and Technology (NIST) [39] has launched a multi-year competition to select the best proposed PQ protocols for Public-key Encryption, Key Establishment, and Digital Signature Algorithms. The winning PQ protocols will become the new standards and will be adopted by governments and industry across the world.

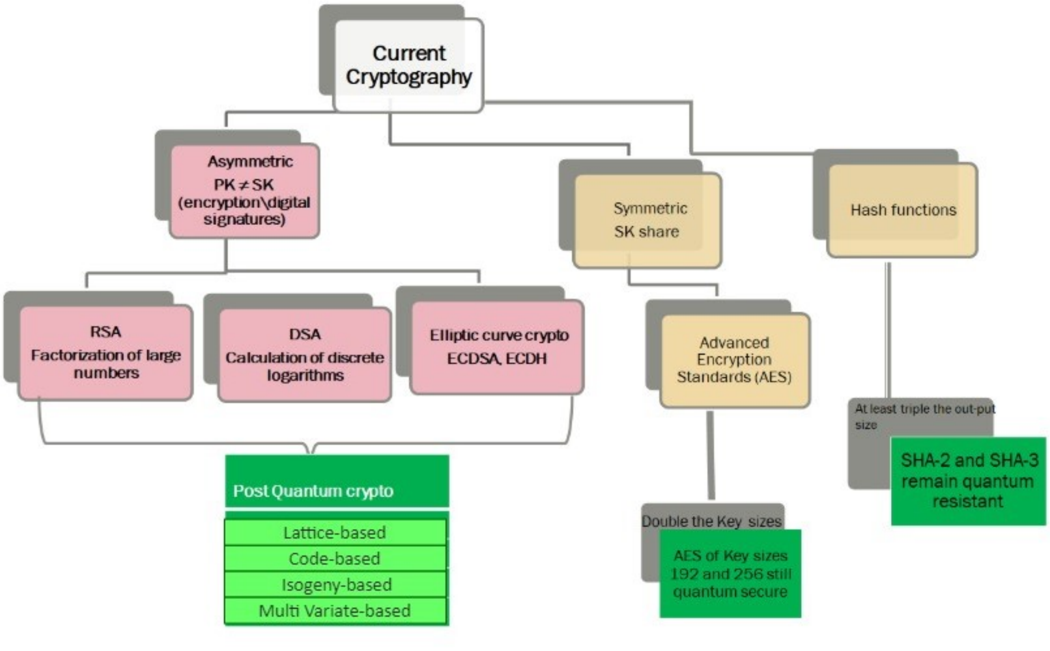

Current Cryptography
Asymmetric PK≠SK (encryption\digital signatures)
Symmetric SK share
Hash functions
RSA Factorization of large numbers
DSA Calculation of discrete logarithms
Elliptic curve crypto ECDSA, ECDH
Advanced Encryption Standards (AES)
At least triple the out-put size
SHA-2 and SHA-3 remain quantum resistant
Post Quantum crypto
Lattice-based
Code-based
Isogeny-based
Multi Variate-based
Double the Key sizes
AES of Key sizes 192 and 256 still quantum secure

Figure 2.4: Overview of the PQ Cryptography.

Although the integration of quantum-safe cryptographic algorithms goes beyond the scope of the ASSURED project, this is something that when realized will affect all ICT systems and supply chains. Thus, in the context of ASSURED, the consortium will investigate (long-term) the use of QR cryptographic functions including Post Quantum (PQ) secure authentication, encryption and signing functions. These primitives form the basic functionality that support more complex operations. Lattice based cryptography have proven to be a flexible tool in constructing cryptographic schemes, with applications ranging from digital signatures to public-key encryption and zero-knowledge proofs, while offering post-quantum security [2, 38]. One expects that as this type of cryptography matures, a more complex lattice based protocols such as latticed based DAA [11, 26], lattice based group signatures [25, 32] and lattice based ring signatures [36] will be developed enough to be integrated in the post quantum ASSURED world as a future research.

Chapter 3

ASSURED Protocols for the DLT Data Management

ASSURED technology modules will provide a secure supply chain data management platform for all stakeholders and parties. ASSURED DLT data management is initially achieved by providing a secure device enrollment that allows ASSURED Blockchain users to verify that only certified edge devices are able to access the Blockchain ledger.

This certification is a two levels process: the first level is achieved through the registration with the Privacy Certification Authority (Privacy CA) that registers the edge device Attestation Key which will be used later to create assurance claims about their states. This registration certification clearly binds the TPM Attestation Key with its Endorsement Key that is originally certified by the TPM manufacturer. Also, user attributes are issued during this secure enrolment phase, by the Privacy Certification Authority (Privacy CA), which then credits the edge device a credential that forwards it to the SCB for verification.

If successful, the SCB will notify the Blockchain Certification Authority (Blockchain CA) which, in turn, will create the appropriate certificate for the device including all verified attributes (as part of the Verifiable Credentials to be issued to the device/user TPM-based Wallet) based on which it can access specific attestation result. The Blockchain certification/credential clearly binds the TPM Attestation Key credential issued by the Privacy CA to the Blockchain keys and user attributes.

Thus, in order to access the ASSURED Blockchain ledger, any edge device should be able to prove that it processes two bounded certifications/ credentials (namely the CA credential and the Blockchain credential) that both link to the TPM Endorsement Key.

ASSURED make use of both private and public ledgers [14]. Hashed attestation results are stored on the ledger whereas the encrypted system traces (control-flow traces or digest lists of loaded binaries) are stored in the ASSURED Data Storage Engine. After querying the encrypted data from the ledger, the decryption of the encrypted data is achieved at the device/user level and is possible only if the set of attributes of the user key matches the attributes of the ciphertext. Access control is a very important requirement for the ASSURED DLT data management. In the ASSURED framework, a TPM is used to support a Key-Policy Attribute-based Encryption (KP-ABE) where the secret key of a Blockchain user and the ciphertext are dependent on the user attributes. In this section we will describe the ABE protocol that will be adopted in the ASSURED framework.

Another DLT data management requirement is the revocation mechanism that provides strong guarantees that when the SCB has initiated and run the protocol to completion, then the associated edge device must have been involved in the protocol instance and correctly received the revocation request and a correct TPM Attestation Key will be revoked at the completion of the protocol. This ensures that any revoked edge devices will not be able to create any further attestation using its revoked Attestation Key.

In what follows, we provide a high level description of the designed ASSURED cryptographic protocols that offer the DLT data management security requirements.

3.1 ASSURED Key Management

Key management is the process of managing cryptographic keys within a cryptosystem. It deals with generating, exchanging, storing, protecting, and replacing keys. A key management system also includes cryptographic protocol designs as defined in [31]. Successful key management is critical to the security of a cryptosystem with the increase dependence on cryptography in everyday applications. It is very important and challenging to keep cryptographic keys safe and secure. If a single key is compromised, this could lead to a massive data leak with the consequential reputational damage and loss of user confidence. Thus, a reliable key management should establish and specify rules to protect its confidentiality, integrity, availability, and authentication of source.

ASSURED framework may require different types of keys to be stored in the device such as Attestation Keys, encryption keys and Blockchain keys. For example, a trusted platform module 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. A Blockchain key is used to access ASSURED Blockchain services and protect the communications between all the edge devices with the backend infrastructure. ASSURED will support all necessary mechanisms for managing the key functionalities and indicate whether there can be a link between some of the keys (derive keys from each other using the trusted component's key hierarchy in order to reduce the key overhead).

To enhance the key management in ASSURED, an important key management requirement, introduced in [13], specifies the usage of the TPM Attestation Key. It states that the AK can only be used if it is binded to the correct state of a device that matches to a specified digest list. If this condition is satisfied, the TPM-based Wallet is then allowed the use of the AK for signing any subsequent attestation processes and traces. This is enforced through the creation of appropriate policy digests that will protect the AK within the wallet. An Attestation Key (AK) may have some attributes such as "Restricted". In this case, AK can either signs messages created by the TPM itself or signs traces provided by an authorized tracer. The next sections will provide concrete protocols that describe the ASSURED key management during the enrollment and revocation phases. Also, a concrete Blockchain key management is described when creating, placing and querying attestations from the ledger.

3.2 Secure Device Enrollment

When enrolling new devices with embedded TPMs, a verifier needs to verify that the given attestation report was created by a genuine TPM even if it is unidentified. To meet this requirement, the TPM has a public and secret AK pair (pk_{AK}, sk_{AK}), which is used for creating attestations either a conventional signature scheme or a Direct Anonymous Attestation (DAA) signature. For the TPM to use its Attestation Key to create signatures, the Attestation Key should have a valid credential issued by the Privacy CA. An attestation Certificate Authority (CA) is involved to authenticate that the Attestation Key AK holder is a genuine TPM in order to issue a credential for the AK. Each TPM has a public and secret Endorsement Key (EK) pair $(pk_{ek}; sk_{ek})$, which is used for an asymmetric encryption scheme. The EK is usually certified by the TPM manufacturer. The credential provider has access to an authentic copy of the public Endorsement Key and its certificate.

The Certification authority CA authenticates the TPM then generates a credential for the TPM Attestation Key. The CA wraps the credential and sends it to the TPM through the device. The TPM unwraps the credential and returns the symmetric decryption key to the device. This TPM operation is called "activate credential". The device decrypts the credential using the decryption key provided by the TPM, then verifies the credential i.e. verifying 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 it needs to generate signatures. The TPM credential is needed whenever a new edge device wants to get access to the Blockchain ledger and executes smart contracts.

Once the device is successfully enrolled with the Privacy CA where it certifies the created Attestation Key (AK), it is then directed to the SCB who verifies that the TPM has a valid CA credential on the TPM public key. If this verification is successful, then any Blockchain user will be able to authenticate and attest the new edge device. The attestation report is then forwarded to the security context broker who verifies the reported attestation result. Upon successful verification, the broker provides the new edge device with Blockchain keys with the necessary attributes that allow the device to access the ledger, securely download and execute the smart contract and upload its attestation results on the Blockchain ledger.

The SCB is also responsible for activating the pseudonyms which are short-term anonymous credentials created by the device and protected under its AK. The registration consists of providing the SCB with unique values that can be used later used to revoke the Attestation Key of a misbehaving device. We call these values revocation hashes. Upon such registration, the device will receive Proof of Registration (PoR), which is provided with any signature from the particular pseudonym. Without PoR, any recipient should disregard the message, as the SCB could not revoke any misbehaving device's key. In case a recipient suspects malicious behavior of a device, it reports the corresponding pseudonym to the SCB.

The detailed protocol to be executed during the device registration phase capturing all the aforementioned operations and interactions with the TPM-based Wallet are described in Section 4.1.

3.3 Attribute-Based Encryption

We will describe ASSURED ABE scheme which is based on [43] with some modifications that allows ASSURED Blockchain users to perform ABE using their embedded TPMs. Our proposed ABE will provide a secure attribute based encryption/decryption scheme for the formatted data via fast and efficient encryption technology where attribute keys are stored into the trusted hardware for extra protection and secure management of various encryption/decryption keys.
Before describing the setup interface we define the following functions that will be used in the proposed ABE protocol.

- $HMAC(M, k_{\text{MAC}})$: A cryptographic hash function used to generate the hash-based message authentication code for a message M according to the integrity key k_{MAC}

- $ENC(M, k_{\text{ENC}})$: A symmetric encryption algorithm, which encrypts the message M with the key k_{ENC}

- $DEC(C, k_{\text{ENC}})$: A symmetric decryption algorithm, which decrypts the cipher-text C with the key k_{ENC}

Next we describe the setup interface initiated by the ASSURED Security Context Broker SCB.

- The authority, which is ASSURED Security Context broker, is responsible on defining the attributes and creating the corresponding attribute keys for each Blockchain user as well as setting up the access control tree. ASSURED SCB follows these steps:

  1. SCB Setup: The attribute space in the system is defined as the universe of attributes $U = 1, 2, \ldots, n$. For each attribute $i \in U$, choose a number t_i uniformly at random from $\mathbb{Z}_q$. The public key of each attribute i is $P_i = t_i \cdot G$, where G is the group generator.
  2. The authority chooses t uniformly at random from $\mathbb{Z}_q$ to be the master (private) key MK, accordingly, the master public key PK is $PK = t \cdot G$. The public parameters are denoted by Params $= PK, P_1, ..., P_U$.
  3. The authority defines, an access tree Γ by creating a public polynomial $q_u(x)$ with order of $d_u - 1$ should be defined for each node u in the access tree Γ in top-down manner such that $q_u(0) = q_{parent}(u)$, where d_u is the threshold of the node u. For the root R of the access tree Γ , set $q_R(0) = t$.
  4. The authority encrypts the attribute based keys for each user under the device's TPM encryption key and send the encrypted attribute keys to the devices.

- In order to describe the steps that must be taken in order to encrypt a clear message, we will assume that an Assured device A wants to encrypt a message to be uploaded on the ASSURED ledger. The steps that a device A must complete are the following:

  1. A device A must create a random secret k and create a shared secret value, which is the result of the scalar multiplication $k \cdot PK$.
  2. Then, A must take the shared secret value $k \cdot PK$ as input data for the Key Derivation Function, KDF. The output of this function is the concatenation of the symmetric encryption key, k_{ENC}, and the MAC key, k_{MAC}. The encrypted message is denoted by $C = ENC(M, k_{\text{ENC}})$ and $MAC = HMAC(M, k_{\text{MAC}})$.
  3. A loads, using its TPM, the public attribute keys needed to perform the encryption of the message M.
  4. With the element k_{ENC} and the clear message, M, A will use the symmetric encryption algorithm, ENC, in order to produce the encrypted message $CM = (\omega, C, MAC, C_i = k \cdot P_i$ for $i \in \omega$ where $\omega \subset U$ is a set of required attributes for the device A.
  5. A uploads the encrypted message together with the access tree Γ.

- Any device B with matching attributes can successfully complete the decryption of device A uploaded on the ledger by performing the following steps:

1. B loads, using its TPM, the attribute keys t_i needed to perform the decryption of the message M.
2. For each C_i on the nodes u of Γ, the device B calculates the corresponding decryption node key denoted by $D : D_u = q_u(0)/t_i$, here, $i = attr(u)$ and t_i is the attribute key assigned to users from $\mathbb{Z}_q$ in Setup phase by the SCB, t_i^{-1} is the inverse element of t_i over finite field $\mathbb{Z}_q$.
3. A device B runs the decryption algorithm DecryptNode(CM, D, u) for a node u in the access tree is defined as a recursive procedure.

   For a leaf node u, where attributes are clearly defined, Let $i = attr(u)$, DecryptNode (CM, D, u) is defined as:
   DecryptNode$(CM, D, u) = D_u \cdot C_i = q_u(0) \cdot t_i^{-1} \cdot k \cdot P_i = q_u(0) \cdot k \cdot G, (i \in \omega)$.

   It should be stated that the output of DecryptNode(CM, D, u) is an element in elliptic curve group or $\perp$.

   Accordingly, for the root node R of the access tree, there should be DecryptNode $(CM, D, R) = q_R(0) \cdot k \cdot G = t \cdot k \cdot G = k \cdot PK$ Using the input $k \cdot PK$ to the derivation function, B the recovers $(K_{\text{KEM}}, K_{\text{MAC}})$.
4. If HMAC$(M, K_{\text{MAC}}) = MAC$, it indicates that message M is correctly decrypted and is not tampered.
5. Device B then uses the recovered symmetric key k_{ENC} to decrypt C and retrieve the original message M.

All details of the newly designed ASSURED ABE scheme along with the interactions between all involved entities are depicted in Section 4.2.

3.4 Property Based Control Flow Attestation & Configuration Integrity Verification

As it pertains to the core provided mechanisms, towards providing operational assurance guarantees through the entire lifecyle of a device, ASSURED leverages novel remote, runtime behavioral attestation services, targeting both the software and hardware layers and covering all phases of a device's execution; from the **trusted boot and integrity measurement of a CPS**, enabling the generation of static, boot-time or load-time evidence of the system's components correct configuration (Configuration Integrity Verification (CIV)), to the runtime behavioral attestation of those safety-critical components of a system providing strong guarantees on the correctness of the control- and information-flow properties (Control-Flow Attestation (CFA)), thus, enhancing the performance and scalability when composing secure systems from potentially insecure components.

In this context, the Verifier is the entity responsible for supporting the different ASSURED attestation schemes [16]. The Verifier, upon reading an attestation policy, initiates a remote attestation process based on the extracted policy, which is populated in the Blockchain infrastructure of ASSURED by the Security Context Broker (SCB) based on the current scheduling policy and risk level assessed per each edge device and the global ASSURED ecosystem. Prior to the device been able to access the attestation contract, it has to first execute the secure device registration process so as to be able to get the appropriate credentials based on which its TPM-based Wallet can create Verifiable Presentations (VPs), including the neceesary attributes, that the SCB will check prior to allowing the device to register to the specific privacte ledger/channel. When registered, the device is then automatically notified for the deployment of any new attestation policy. In the case of a policy destined for the specific device, it is sent to the device Attestation Agent, which allows invoking the requested attestation task: either CFA or CIV. Additional parameters are also read from the Blockchain, including a fresh nonce and parameters that are required for the attestation task, such as a process identifier that the tracer will trace for the CFA attestation scheme. The attestation agent sends the tracer a request to compute the trace according to the extracted parameters. The tracer computes a trace, signs it, and passes it back to the attestation agent for verification.

All details for the entire attestation-related workflow of actions, as it pertains to the secure attestation evidence management and sharing, are depicted in Section 4.3.

3.5 Pseudonyms and Revocation in ASSURED

ASSURED designed a revocation protocol that enables successful de-activation of a misbehaving device's credentials without disclosing its identity [34]. Before starting describing the protocol we highlight the following assumptions, as also defined in D3.2 [16]:

- It is assumed that already in place a misbehavior detection mechanism equipped with the necessary rules and policies for detecting any malevolent actions from insider attackers. ASSURED focuses on how to revoke the attacker's credentials once a misbehavior has been detected.
- The device to be revoked was successfully enrolled with the Privacy CA and SCB for creating a number of pseudonyms, short-term anonymous credentials that can be used for enhanced privacy (see Section 3.2 for more details). This entity is important, as it should not be possible for a device to successfully use a pseudonym unless it has registered it with the SCB.
- Upon receiving the revocation message from the SCB, all devices forward the exact revocation message them to their corresponding TPM-based Wallet. Here we eliminate any possible man in the middle attacks that may let the device block or alter the content of the revocation message.

We start by introducing the SCB that notices out misbehaving devices from the system within the revocation domain that it's managing. The SCB does not perform any pseudonym resolution to discover the identity of the misbehaving platform; in contrast with this process that needs to be performed with traditional Public key Infrastructures (PKI) and can breach the privacy of shunned out devices. Instead, it starts the revocation protocol by creating a signed revocation message using its secret key and broadcasts this to all devices, containing the public pseudonym key that needs to be revoked. All devices receive the revocation message, and forward them to their corresponding TPM-based Wallet. The protocol differentiates between two kinds of revocations: **soft- and hard-revocation**. Soft revocation means the SCB revokes a specific pseudonym that was used for signing the message based on which the misbehavior policy violation was detected, while hard revocation means revoking all pseudonyms associated with a specific platform, thus, not allowing it to further take part in the overall system as an authenticated participant (basically, de-activating all pseudonyms created under a specific DAA AK).

The details of the ASSURED revocation protocol and interactions between all participating entities are depicted in Section 4.4.

Chapter 4

Secure Data Management Implementation Diagrams

This Chapter presents the concrete cryptographic sequence diagrams of the protocols explained in Chapter 3. These diagrams explain the sequence of cryptographic actions as will be executed by the different ASSURED entities.

4.1 Secure Device Enrollment Sequence Diagram

During the registration and enrollment phase, the primary goal of a device is to receive a keypair so as to then be able to securely send (attestation) data to the Blockchain. To achieve this, it must prove that it holds an Attestation Key that will only provide signatures under a correct configuration (Figure 4.1).

To create such an Attestation Key, technically an ECC DAA key, the device 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. In practical terms, it's a `PolicyAuthorize` policy.

The device sends the key to the Privacy CA for verification when the key is created. Note that this is not only the cryptographic public key. This TPM-specific public key holds extra information, s.a., operational requirements, technical attributes, etc. As the Privacy CA verifies the data, it returns a challenge credential as part of the DAA protocol to the device. However, the device cannot yet use the key to continue the DAA join process. Recall that the SCB must authorize all policies for cryptographic operations.

Therefore, the Privacy CA sends information to the SCB needed to generate policies. This information consists of a signature, confirming the key contains the right policy and attributes, and the local tracers public key. The key's primary operation is to sign data; hence the SCB must create an authorization for this. An authorization essentially consists of a signed policy, and the policy itself can be based on multiple policy requirements. For signing, these requirements build on a particular configuration represented by the local TC's PCR registers *and* a signature over the session nonce from the tracers key. This ensures that the TPM can only execute signing operation if the platform is configured correctly and the tracer authorized it - guaranteeing unauthorized entities cannot use the key (Figure 4.2).

Symbol	Description
G	Group generator
q	A large prime number
$\mathbb{Z}_q$	A finite integer field, whose elements set is $\{0, 1, ..., q-1\}$
MK	The master private key of the ABE scheme
PK	The master public key of the ABE scheme
sk_{ek}/sk_{ek}	A Endorsement Key-pair
pk_{scb}/sk_{scb}	A Security Context Broker Key-pair
pk_{trc}/sk_{trc}	A Tracer Key-pair
pk_{bc}/sk_{bc}	A Blockchain Key-pair
pk_{RA}/sk_{RA}	A Revocation Authority Key-pair
pk_{ak}/sk_{ak}	An Attestation Key-pair
pk_{pca}/sk_{pca}	A Privacy Certification Authority Key-pair
Kh_x	A Key Handle pointing to the loaded key ”x” in the TPM
σ_x	A Cryptographic signature identified by ”x”
$\mathcal{I}$	A PCR Selection.
$\mathcal{N}$	A Collection of Traces.
R	A Revocation Index of Traces.
cre	DAA Challenge Credential
F_{cre}	Full (encrypted) DAA Credential
DAA_{cre}	Full (decrypted) DAA Credential
$\mathcal{C}$	Digest representing a state of a device
CC	TPM Command Code
σ_x	Cryptographic signature
a_{sign}	Authorized Digest for Signing
a_{comm}	Authorized Digest for Committing
s_x	Session Handle identified by ”x”
h_x	Hash Digest identified by ”x”
$'\, n_x$	Nonce identified by ”x”
t_x	TPM Ticket identified by ”x”
$H(x)$	Hash of ”x”
N_{DAA}	Prepared String for DAA hash / Sign og Traces
k_{x_n}	TPM Name of Key identified by ”x”
C_{data}	TPM Commit Data provided by TPM2_Commit
CP	A string used in the join phase of DAA
H_H	Hard Revocation Hash
H_S	Soft Revocation Hash
PoR	Proof of Registration

Table 4.1: Notations Used in the Enrollment, ABE, CFA and Revocation Protocols

The second operation the key is used for is executing the first part of the cryptographic signature; a TPM2_Commit. As this command on its own is unable to provide signatures, the SCB authorizes all executions of this command. The two authorizations are then sent to the device, which uses the TPM to verify the signature and obtain a verification ticket. This ticket proves to the TPM, in future contexts, that the authorizations have been verified at an earlier state and are valid.

After this, the device continues the JOIN process and activates the credential to obtain the credential key, which is used in a later stage. The device now shows the authorization to commit to the TPM and executes the command, retrieving the commit data. The host then prepares the response string to the SCB, as per the DAA Protocol, using this information obtained by activating the credential and committing. Because the Attestation Key is restricted, the TPM must hash the prepared string and not the device.

To sign the hashed value, the device must satisfy the sign authorization. This is done by loading

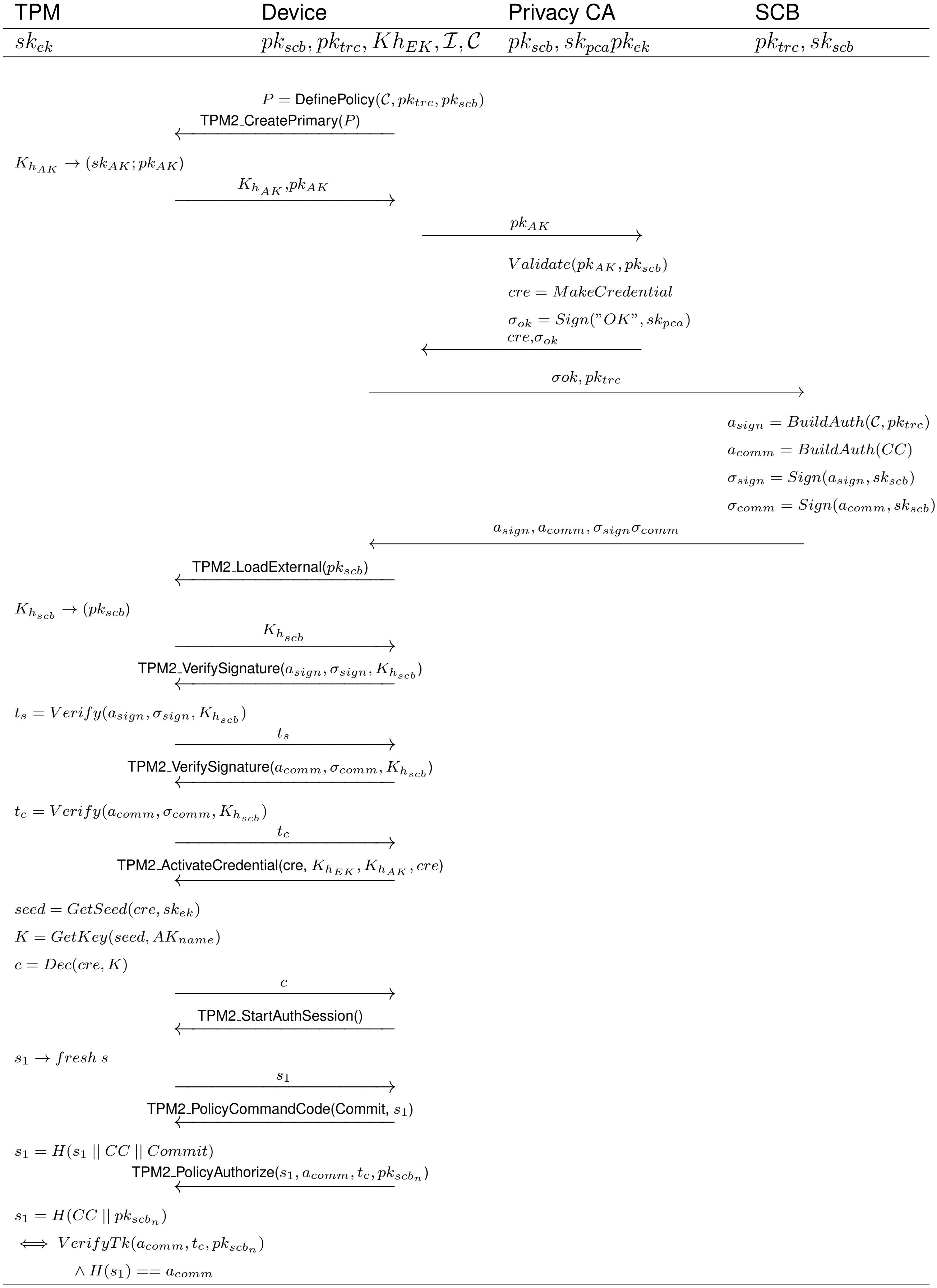
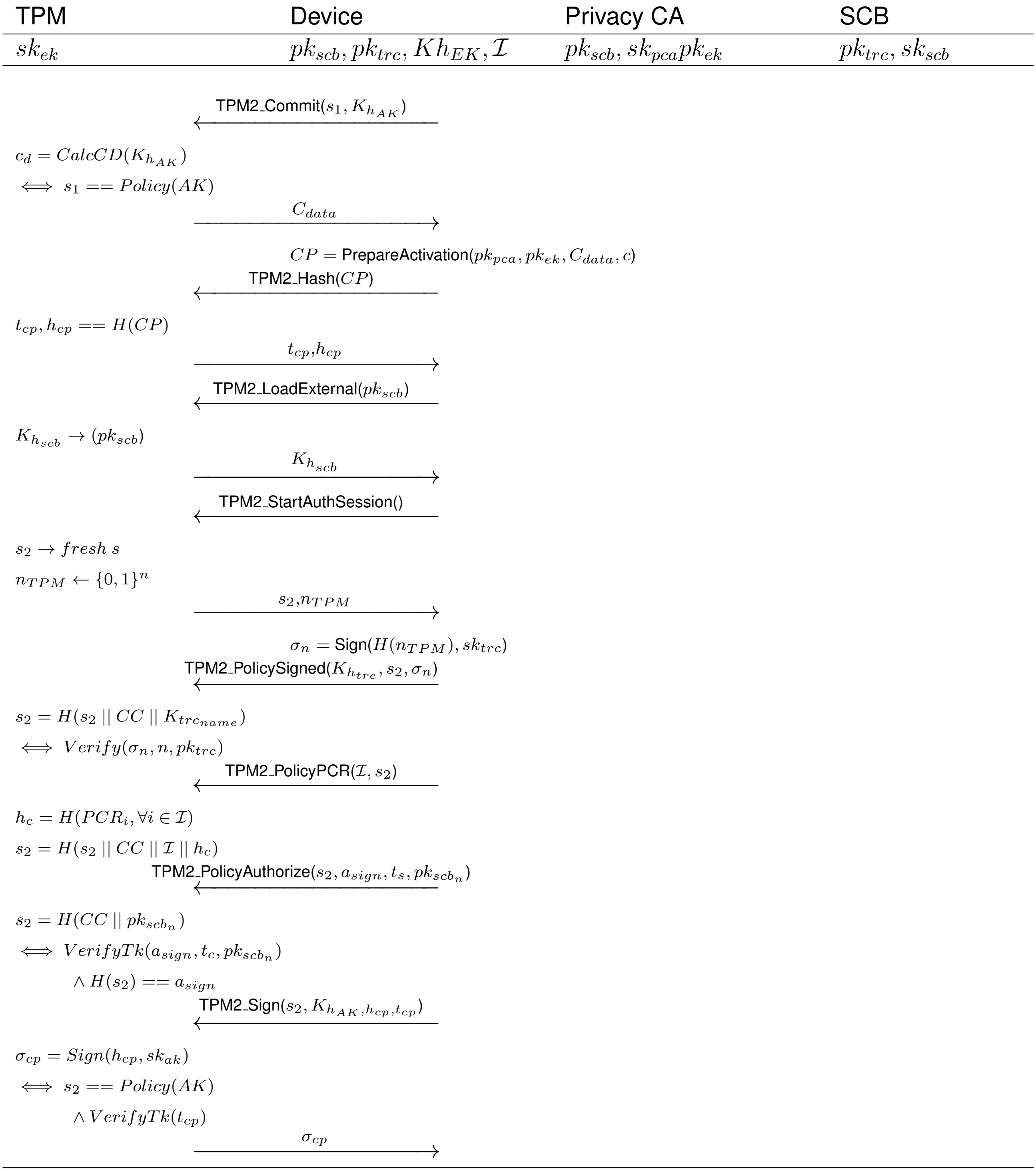

TPM
Device
Privacy CA
SCB
sk_{ek}
$pk_{scb}, pk_{trc}, Kh_{EK}, \mathcal{I}, \mathcal{C}$
$pk_{scb}, sk_{pca}pk_{ek}$
pk_{trc}, sk_{scb}
$P =$ DefinePolicy$(\mathcal{C}, pk_{trc}, pk_{scb})$
TPM2_CreatePrimary(P)
$K_{h_{AK}} \rightarrow (sk_{AK}; pk_{AK})$
$K_{h_{AK}}, pk_{AK}$
pk_{AK}
$Validate(pk_{AK}, pk_{scb})$
$cre = MakeCredential$
$\sigma_{ok} = Sign(”OK”, sk_{pca})$
cre, σ_{ok}
$\sigma ok, pk_{trc}$
$a_{sign} = BuildAuth(\mathcal{C}, pk_{trc})$
$a_{comm} = BuildAuth(CC)$
$\sigma_{sign} = Sign(a_{sign}, sk_{scb})$
$\sigma_{comm} = Sign(a_{comm}, sk_{scb})$
$a_{sign}, a_{comm}, \sigma_{sign}\sigma_{comm}$
TPM2_LoadExternal(pk_{scb})
$K_{h_{scb}} \rightarrow (pk_{scb})$
$K_{h_{scb}}$
TPM2_VerifySignature$(a_{sign}, \sigma_{sign}, K_{h_{scb}})$
$t_s = Verify(a_{sign}, \sigma_{sign}, K_{h_{scb}})$
t_s
TPM2_VerifySignature$(a_{comm}, \sigma_{comm}, K_{h_{scb}})$
$t_c = Verify(a_{comm}, \sigma_{comm}, K_{h_{scb}})$
t_c
TPM2_ActivateCredential(cre, $K_{h_{EK}}, K_{h_{AK}}, cre$)
$seed = GetSeed(cre, sk_{ek})$
$K = GetKey(seed, AK_{name})$
$c = Dec(cre, K)$
c
TPM2_StartAuthSession()
$s_1 \rightarrow fresh\ s$
s_1
TPM2_PolicyCommandCode(Commit, s_1)
$s_1 = H(s_1 \mid\mid CC \mid\mid Commit)$
TPM2_PolicyAuthorize$(s_1, a_{comm}, t_c, pk_{scb_n})$
$s_1 = H(CC \mid\mid pk_{scb_n})$
$\iff VerifyTk(a_{comm}, t_c, pk_{scb_n})$
$\wedge\ H(s_1) == a_{comm}$

Figure 4.1: Secure Enrolment pt 1



TPM	Device	Privacy CA	SCB
sk_{ek}	$pk_{scb}, pk_{trc}, Kh_{EK}, \mathcal{I}$	$pk_{scb}, sk_{pca}pk_{ek}$	pk_{trc}, sk_{scb}

$\xleftarrow{\text{TPM2\_Commit}(s_1, K_{h_{AK}})}$

$c_d = CalcCD(K_{h_{AK}})$

$\iff s_1 == Policy(AK)$

$\xrightarrow{C_{data}}$

$CP =$ PrepareActivation$(pk_{pca}, pk_{ek}, C_{data}, c)$

$\xleftarrow{\text{TPM2\_Hash}(CP)}$

$t_{cp}, h_{cp} == H(CP)$

$\xrightarrow{t_{cp}, h_{cp}}$

$\xleftarrow{\text{TPM2\_LoadExternal}(pk_{scb})}$

$K_{h_{scb}} \rightarrow (pk_{scb})$

$\xrightarrow{K_{h_{scb}}}$

$\xleftarrow{\text{TPM2\_StartAuthSession()}}$

$s_2 \rightarrow fresh\ s$

$n_{TPM} \leftarrow \{0,1\}^n$

$\xrightarrow{s_2, n_{TPM}}$

$\sigma_n =$ Sign$(H(n_{TPM}), sk_{trc})$

$\xleftarrow{\text{TPM2\_PolicySigned}(K_{h_{trc}}, s_2, \sigma_n)}$

$s_2 = H(s_2 \mid\mid CC \mid\mid K_{trc_{name}})$

$\iff Verify(\sigma_n, n, pk_{trc})$

$\xleftarrow{\text{TPM2\_PolicyPCR}(\mathcal{I}, s_2)}$

$h_c = H(PCR_i, \forall i \in \mathcal{I})$

$s_2 = H(s_2 \mid\mid CC \mid\mid \mathcal{I} \mid\mid h_c)$

$\xleftarrow{\text{TPM2\_PolicyAuthorize}(s_2, a_{sign}, t_s, pk_{scb_n})}$

$s_2 = H(CC \mid\mid pk_{scb_n})$

$\iff VerifyTk(a_{sign}, t_c, pk_{scb_n})$

$\wedge H(s_2) == a_{sign}$

$\xleftarrow{\text{TPM2\_Sign}(s_2, K_{h_{AK}, h_{cp}, t_{cp}})}$

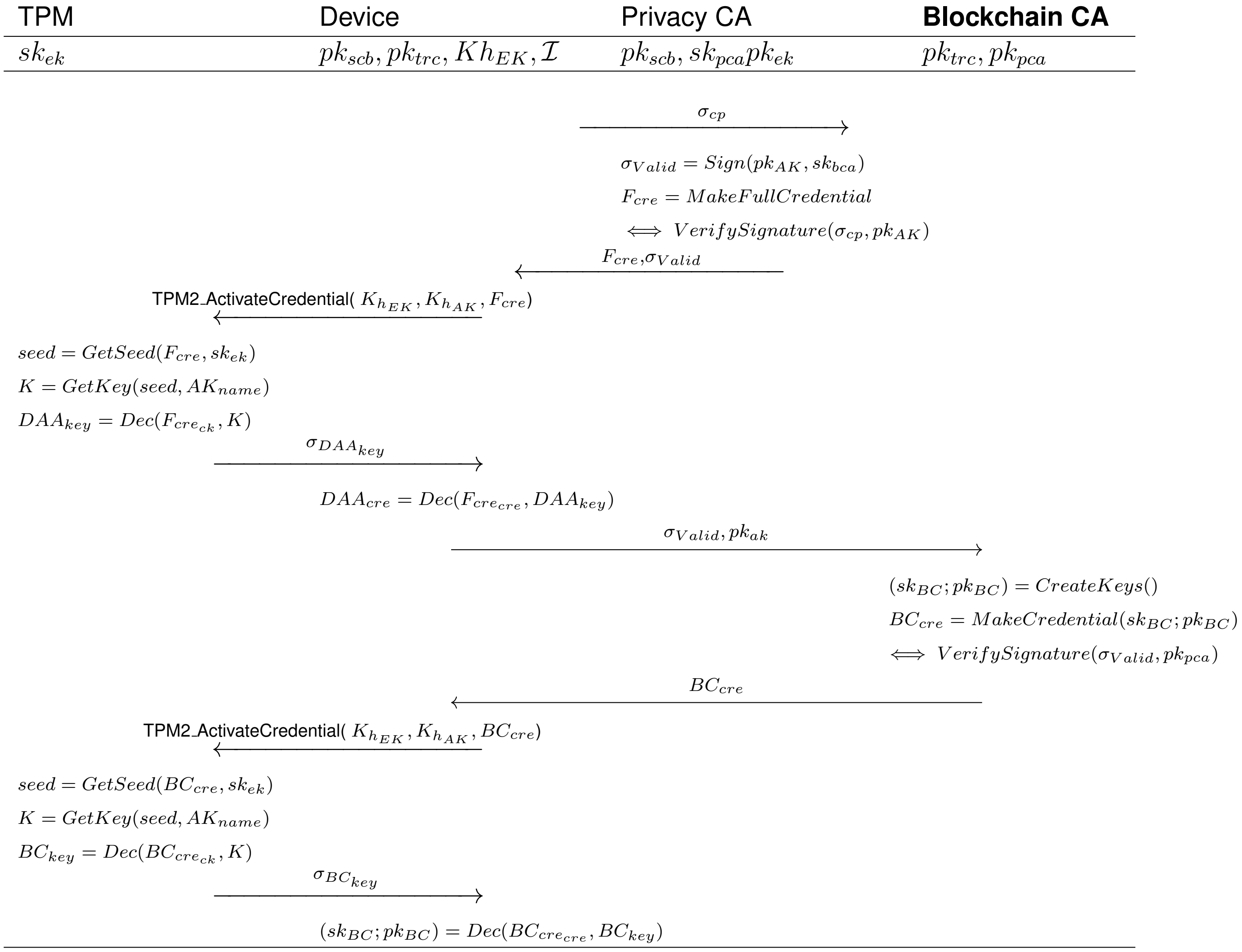
$\sigma_{cp} = Sign(h_{cp}, sk_{ak})$

$\iff s_2 == Policy(AK)$

$\wedge VerifyTk(t_{cp})$

$\xrightarrow{\sigma_{cp}}$

Figure 4.2: Secure Enrolment pt 2



TPM	Device	Privacy CA	**Blockchain CA**
sk_{ek}	$pk_{scb}, pk_{trc}, Kh_{EK}, \mathcal{I}$	$pk_{scb}, sk_{pca}pk_{ek}$	pk_{trc}, pk_{pca}

$\xrightarrow{\sigma_{cp}}$

$\sigma_{Valid} = Sign(pk_{AK}, sk_{bca})$

$F_{cre} = MakeFullCredential$

$\iff VerifySignature(\sigma_{cp}, pk_{AK})$

$\xleftarrow{F_{cre}, \sigma_{Valid}}$

TPM2_ActivateCredential($K_{h_{EK}}, K_{h_{AK}}, F_{cre}$)

$\longleftarrow$

$seed = GetSeed(F_{cre}, sk_{ek})$

$K = GetKey(seed, AK_{name})$

$DAA_{key} = Dec(F_{cre_{ck}}, K)$

$\xrightarrow{\sigma_{DAA_{key}}}$

$DAA_{cre} = Dec(F_{cre_{cre}}, DAA_{key})$

$\xrightarrow{\sigma_{Valid}, pk_{ak}}$

$(sk_{BC}; pk_{BC}) = CreateKeys()$

$BC_{cre} = MakeCredential(sk_{BC}; pk_{BC})$

$\iff VerifySignature(\sigma_{Valid}, pk_{pca})$

$\xleftarrow{BC_{cre}}$

TPM2_ActivateCredential($K_{h_{EK}}, K_{h_{AK}}, BC_{cre}$)

$\longleftarrow$

$seed = GetSeed(BC_{cre}, sk_{ek})$

$K = GetKey(seed, AK_{name})$

$BC_{key} = Dec(BC_{cre_{ck}}, K)$

$\xrightarrow{\sigma_{BC_{key}}}$

$(sk_{BC}; pk_{BC}) = Dec(BC_{cre_{cre}}, BC_{key})$

Figure 4.3: Secure Enrolment pt 3

the Tracers public key into the TPM, starting a new session that returns nonce. The device, or more specifically, the tracer, must then sign this nonce to allow the join to continue. This is proved to be done using `PolicySigned`. To prove the configuration of the platform, the device executes the `PolicyPCR` command. If all policies are satisfied correctly, the sessions policy digest would now match the signed authorization for signature operations, and the sign is executed.

This signature is sent to the Privacy CA that verifies the signature, and returns a full credential (per DAA protocol), a signature over the attestations key public key (to prove it is authorized). The device uses the TPM to activate the credential and get the symmetric key to decrypt the DAA Credential.

To obtain the Blockchain keypair (Figure 4.3), the device shares the public key and the Privacy CA's signature authorizing it with the SCB. The SCB builds a keypair and packs it in a credential that can only be activated in the correct TPM, having both the unique endorsement key and the Attestation Key. The device can now activate the credential, and decrypt the Blockchain keypair. At any point in time, the SCB can authorize new signing requirements, e.g., in case of a platform update.

4.2 Attribute-Based Encryption Sequence Diagram

ASSURED TPM wallet supports ABE that is used in to ensure legitimate attribute-based access control to sensitive encrypted data. A trusted authority which can be many entities, generates the user attribute keys using the authority master key MK as shown in Figure 4.4. The trusted authority then sends the encrypted attribute symmetric keys together with the attribute policies to the Blockchain TPM wallet. The TPM stores the encrypted attribute symmetric keys and, using its own asymmetric decryption keys, decrypts the corresponding attribute symmetric keys and outputs them (from inside TPM) to the edge device/host when requested in order to decrypt the cyphertext stored on the ledger.

The protocol consists of three interfaces, namely setup, encrypt and decrypt interface. In th setup interface, the authority defines the attribute space $U = 1, 2, \ldots, n$ and creates the attribute key $(t_i, P_i = t_i \cdot G)$ that corresponds to each attribute $i \in U$. It also defines as the access control tree Γ that shows the list of attributes required for each access control task. The authority chooses a secret element t and sets its private/public key pair as $(MK = t, PK = t \cdot G)$. The authority encrypts the attribute based keys for each user under the TPM wallet Endorsement key and sends the encrypted attribute keys to the device.

In the encryption interface, a device A (which is the encryptor) creates a random secret k and a shared secret value $k \cdot PK$. Then, A must take the shared secret value $k \cdot PK$ as input data for the Key Derivation Function, KDF that outputs k_{ENC}, and the MAC key, k_{MAC}. The encrypted message $CM = (\omega, C, MAC, C_i = k \cdot P_i$ for $i \in \omega$ where $\omega \subset U$ is a set of required attributes for the device A and $C = ENC(M, k_{\text{ENC}})$ and $MAC = HMAC(M, k_{\text{MAC}})$.

In the decryption interface, a device B(which is the decryptor) loads, using its TPM wallet, the attribute keys t_i needed to perform the decryption of the message M. For each C_i on the nodes u of Γ, the device B calculates the corresponding decryption node key denoted by $D : D_u = q_u(0)/t_i$, here, $i = attr(u)$, t_i^{-1} is the inverse element of t_i over finite field $\mathbb{Z}_q$. B runs recursively from the lower leave to the root of the tree the algorithm DecryptNode(CM, D, u) for a node u defined by:
DecryptNode$(CM, D, u) = D_u \cdot C_i = q_u(0) \cdot t_i^{-1} \cdot k \cdot P_i = q_u(0) \cdot k \cdot G, (i \in \omega)$.

Device A	SCB	Device B
	$U = 1, 2, \ldots, n$ $\forall\ i \in U$ $t, t_i \leftarrow\ Z_q$ $P_i = t_i \cdot G$ $MK = t$, $PK = t \cdot G$ Γ, $q_u(x)$, $q_R(0) = t$	
	$\xleftarrow{(P_i, t_i)\ \forall i\ \in U_A}$	$\xrightarrow{(P_i, t_i)\ \forall i\ \in U_B}$
$k \leftarrow \mathbb{Z}_q$ TPM2_Load(P_i) $KDF(k.PK) = (k_{\text{ENC}}, k_{\text{MAC}})$ $C = ENC(M, k_{\text{ENC}})$ $MAC = HMAC(M, k_{\text{MAC}})$ $CM = (\omega, C, MAC, C_i = k \cdot P_i)$ Upload CM to the ledger		
		TPM2_Load(t_i) $\forall\ (C_i, u)$ of Γ $D_u = q_u(0)/t_i$ DecryptNode$(CM, D, u) = D_u \cdot C_i$ $= q_u(0) \cdot t_i^{-1} \cdot k \cdot P_i$ $= q_u(0) \cdot k \cdot G, (i \in \omega)$ DecryptNode$(CM, D, R) = q_R(0) \cdot k \cdot G$ $= t \cdot k \cdot G = (K_{\text{KEM}}, K_{\text{MAC}})$ If $HMAC(M, K_{\text{MAC}}) = MAC$ Decrypt M using k_{ENC}

Figure 4.4: ASSURED ABE

For the root node R of the access tree, there should be DecryptNode$(CM, D, R) = q_R(0) \cdot k \cdot G = t \cdot k \cdot G = k \cdot PK$.

Using this input in the KDF, the decryptor will recover $(K_{\text{KEM}}, K_{\text{MAC}})$. If HMAC$(M, K_{\text{MAC}}) = MAC$, it indicates that message M is correctly decrypted and is not tampered. Device B then uses the recovered symmetric key k_{ENC} to decrypt C and retrieve the original message M.

4.3 Property Based Control Flow Attestation & Configuration Integrity Verification Sequence Diagram

When it comes to attestation, the process is initiated by the Verifier. It pulls the policy from the Blockchain and obtains a nonce. To prove to the Prover that the request is from a valid Verifier, it will sign this nonce using its private key. It will then contact the Prover with the signed nonce and properties from the policy that should be attested. Assuming the device can verify the signature, the device instructs the trusted tracer to trace the properties defined, obtaining the trace-set $\mathcal{N}$ which is to be signed by the Attestation Key (Figure 4.5). Recall that to use the Attestation Key, an authorized policy by the SCB must be satisfied. The first part of the signature is the Commit. To allow for the Commit, we must satisfy the commit authorization from the SCB, which consists of limiting ourselves to execute that command using `TPM2_PolicyCommandCode`. We then verify that our session matches the authorized digest and execute the Commit. Again, recall the Attestation Key is restricted, so we ask the TPM to hash the traces. Now, the second authorized policy from the SCB must be satisfied (Figure 4.6). To make sure we start from a clean session digest, we execute `PolicyRestart` which resets the digest but keeps the nonce. To allow for a signing operation, the tracer computes a local authorization, essentially a concatenation of the allowed command, the parameters, and the session nonce. This is sent to the TPM in `PolicySigned` which verifies the signature and updates the session. To completely satisfy the SCB policy, we instruct the TPM to read the PCR registers into the session as well. Now the `PolicyAuthorize` command is executed to ensure the current session digest matches the authorized signing policy, and the final signing operation can happen (if the correct parameters are provided). The resulting signature and digest of traces are sent to the Verifier for verification.

4.4 Revocation

in order for the correct execution of the revocation protocol, devices must register pseudonyms with proper revocation authorities, in our case, the Security Context Broker. To do so (Figure 4.7), the device defines a policy which binds the use of a pseudonym to a particular bit (i) in a revocation index R. This is described in detail in D3.2 [16]. Using this policy, the device instructs the TPM to create a pseudonym under the Attestation Key. The public part of the pseudonym and the unique revocation hashes (calculated prior to pseudonym creation) are then sent to the SCB who returns a Proof of Registration (PoR), needed for communication using pseudonyms.

For revocation the SCB locates the revocation hashes for the pseudonym to revoke. Depending on the type of revocation (soft or hard), the SCB sign the desired digest (h_x). The digest and signature are sent to the device, which locates the bit index assigned to this revocation hash. Before using this (or these) bit(s) in a TPM command, the policy for writing to R must be satisfied. This consists of ensuring only correct bits can be set and that it indeed was authorized by a valid SCB. Again, we refer to deliverable 3.2 for details.

Finally, the device executes `TPM2_NVSetBits` with a bit mask that will set these bits in R, effectively revoking one or more pseudonyms.

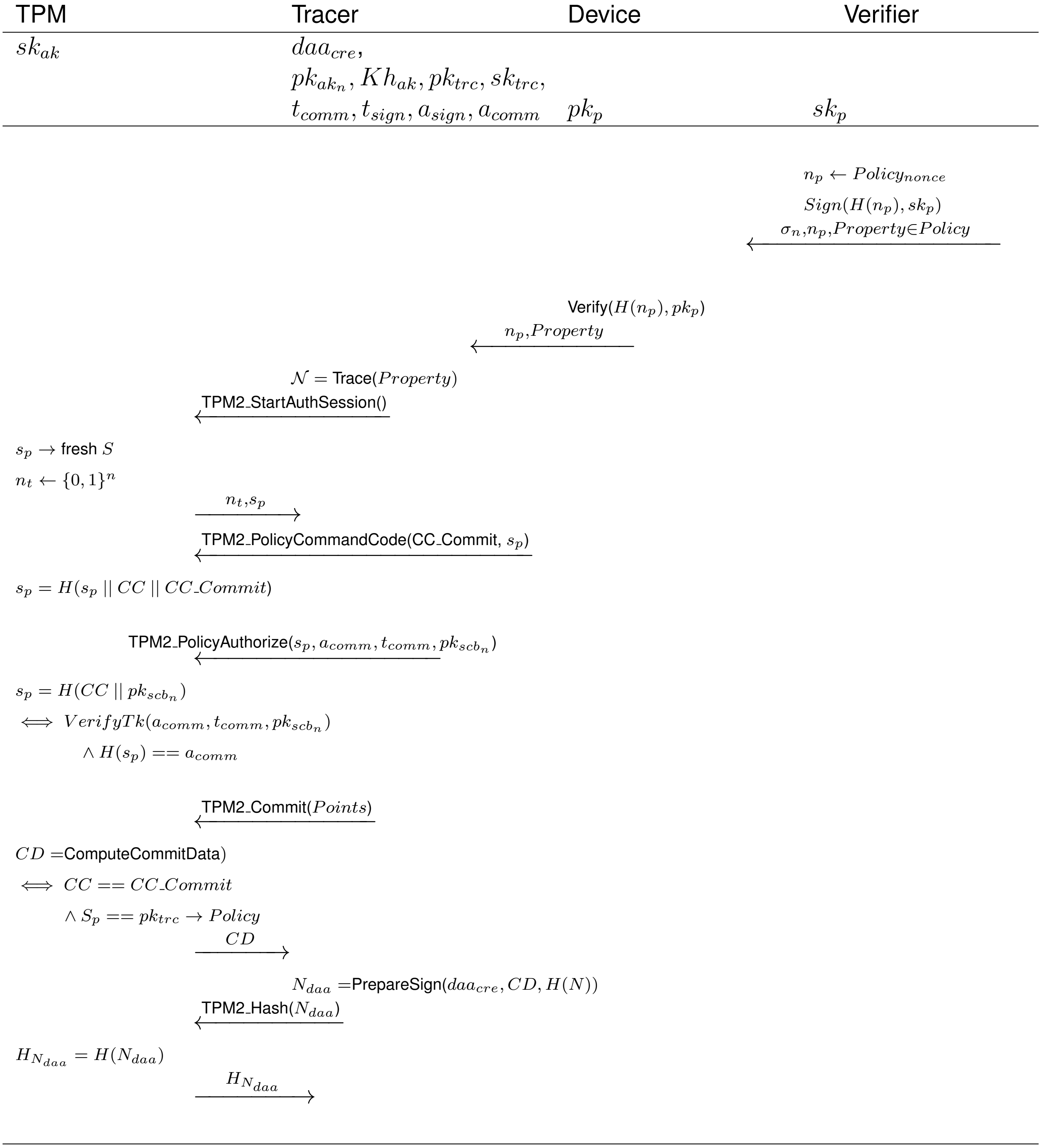

TPM
Tracer
Device
Verifier
sk_{ak}
daa_{cre},
$pk_{ak_n}, Kh_{ak}, pk_{trc}, sk_{trc}$,
$t_{comm}, t_{sign}, a_{sign}, a_{comm}$
pk_p
sk_p
$n_p \leftarrow Policy_{nonce}$
$Sign(H(n_p), sk_p)$
$\sigma_n, n_p, Property \in Policy$
Verify($H(n_p), pk_p$)
$n_p, Property$
$\mathcal{N}$ = Trace($Property$)
TPM2_StartAuthSession()
$s_p \rightarrow$ fresh S
$n_t \leftarrow \{0,1\}^n$
n_t, s_p
TPM2_PolicyCommandCode(CC_Commit, s_p)
$s_p = H(s_p \mid\mid CC \mid\mid CC\_Commit)$
TPM2_PolicyAuthorize($s_p, a_{comm}, t_{comm}, pk_{scb_n}$)
$s_p = H(CC \mid\mid pk_{scb_n})$
$\iff VerifyTk(a_{comm}, t_{comm}, pk_{scb_n})$
$\wedge H(s_p) == a_{comm}$
TPM2_Commit($Points$)
CD =ComputeCommitData)
$\iff CC == CC\_Commit$
$\wedge S_p == pk_{trc} \rightarrow Policy$
CD
N_{daa} =PrepareSign($daa_{cre}, CD, H(N)$)
TPM2_Hash(N_{daa})
$H_{N_{daa}} = H(N_{daa})$
$H_{N_{daa}}$

Figure 4.5: Control-Flow Attestation Command Sequence, pt. 1

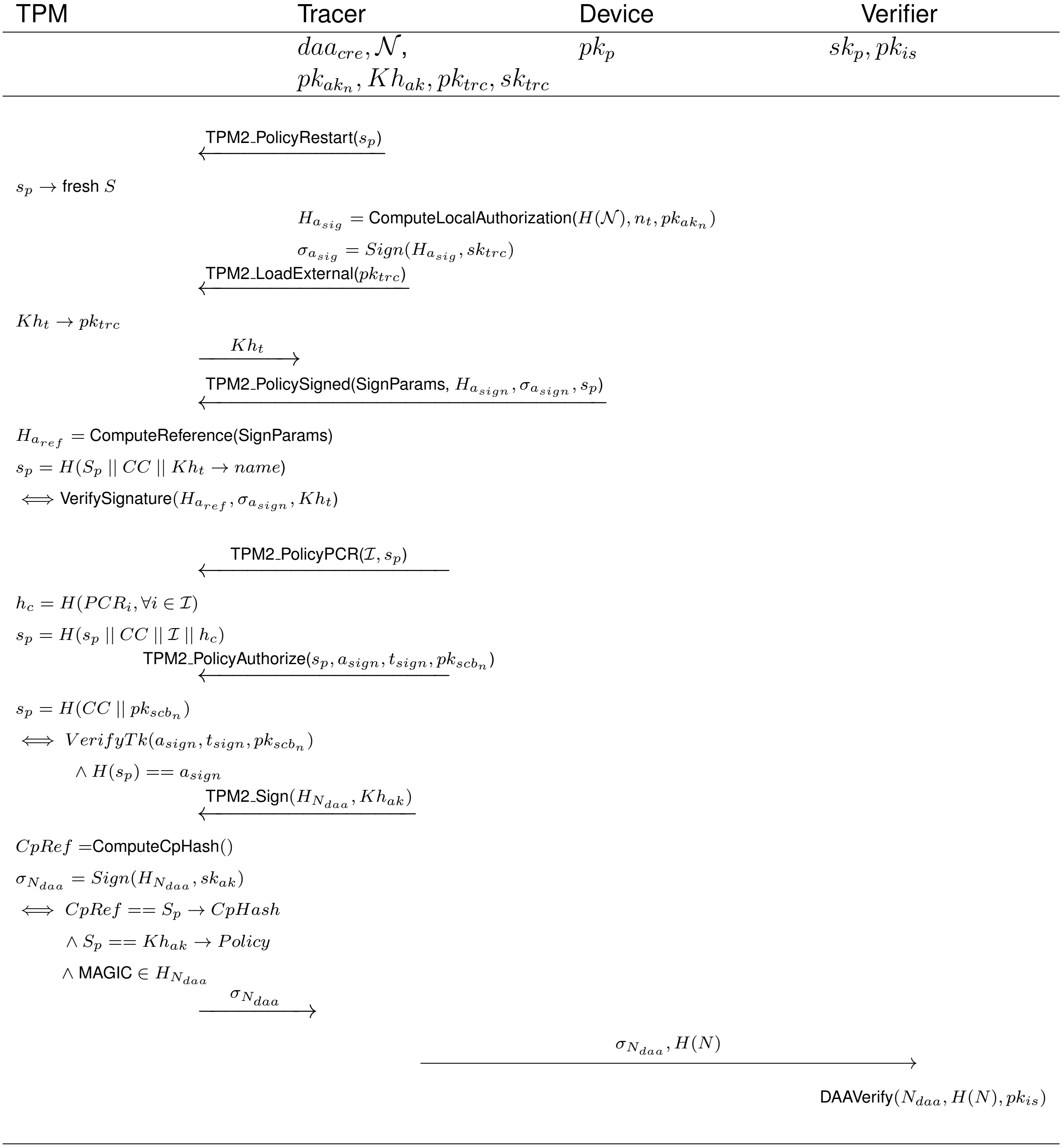

TPM
Tracer
Device
Verifier
$daa_{cre}, \mathcal{N},$
$pk_{ak_n}, Kh_{ak}, pk_{trc}, sk_{trc}$
pk_p
sk_p, pk_{is}
TPM2_PolicyRestart(s_p)
$s_p \to$ fresh S
$H_{a_{sig}} =$ ComputeLocalAuthorization($H(\mathcal{N}), n_t, pk_{ak_n}$)
$\sigma_{a_{sig}} = Sign(H_{a_{sig}}, sk_{trc})$
TPM2_LoadExternal(pk_{trc})
$Kh_t \to pk_{trc}$
Kh_t
TPM2_PolicySigned(SignParams, $H_{a_{sign}}, \sigma_{a_{sign}}, s_p$)
$H_{a_{ref}} =$ ComputeReference(SignParams)
$s_p = H(S_p \mid\mid CC \mid\mid Kh_t \to name)$
$\iff$ VerifySignature($H_{a_{ref}}, \sigma_{a_{sign}}, Kh_t$)
TPM2_PolicyPCR($\mathcal{I}, s_p$)
$h_c = H(PCR_i, \forall i \in \mathcal{I})$
$s_p = H(s_p \mid\mid CC \mid\mid \mathcal{I} \mid\mid h_c)$
TPM2_PolicyAuthorize($s_p, a_{sign}, t_{sign}, pk_{scb_n}$)
$s_p = H(CC \mid\mid pk_{scb_n})$
$\iff VerifyTk(a_{sign}, t_{sign}, pk_{scb_n})$
$\wedge H(s_p) == a_{sign}$
TPM2_Sign($H_{N_{daa}}, Kh_{ak}$)
$CpRef =$ComputeCpHash()
$\sigma_{N_{daa}} = Sign(H_{N_{daa}}, sk_{ak})$
$\iff CpRef == S_p \to CpHash$
$\wedge S_p == Kh_{ak} \to Policy$
$\wedge$ MAGIC $\in H_{N_{daa}}$
$\sigma_{N_{daa}}$
$\sigma_{N_{daa}}, H(N)$
DAAVerify($N_{daa}, H(N), pk_{is}$)

Figure 4.6: Control-Flow Attestation Command Sequence, pt. 2

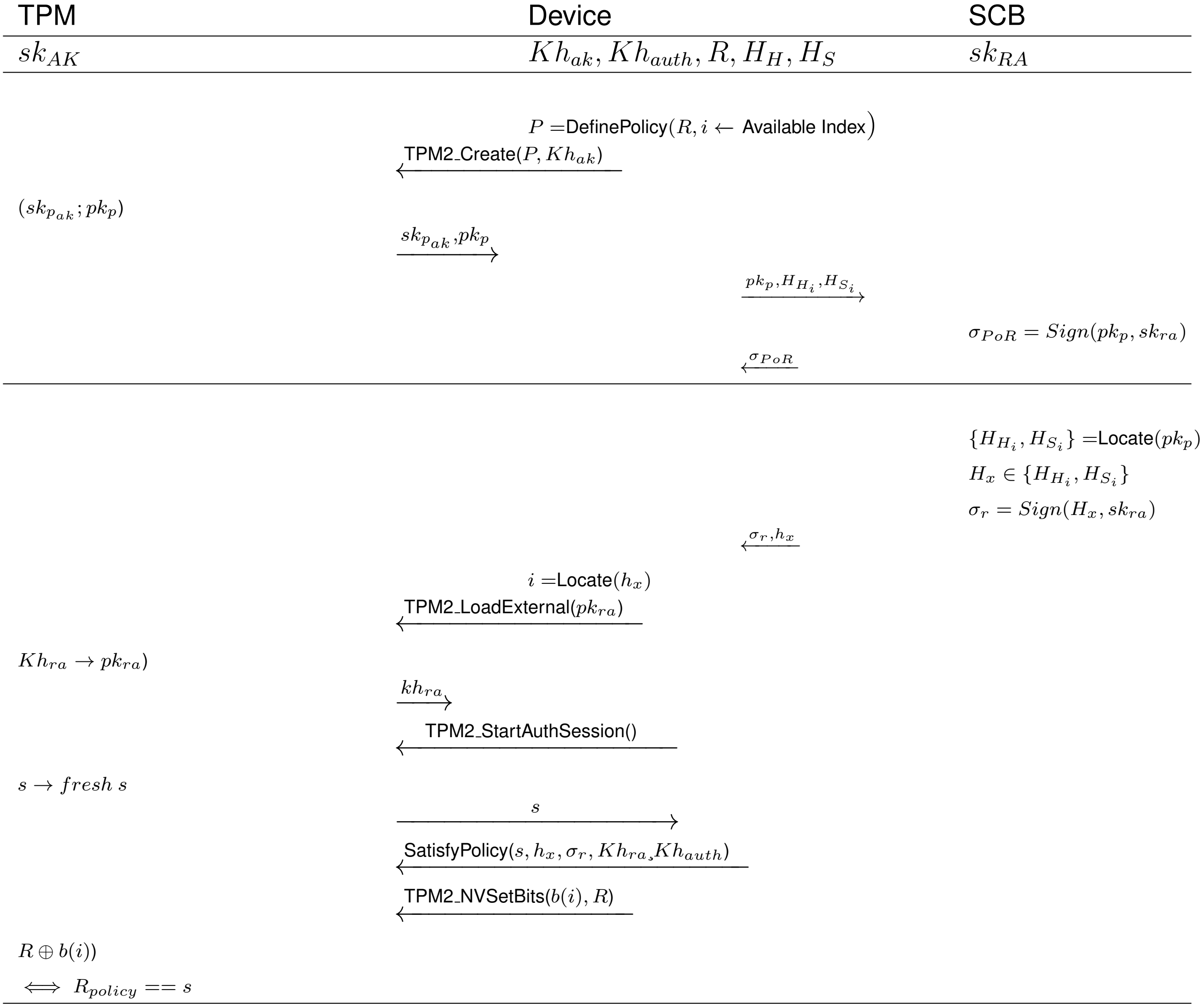

TPM
Device
SCB
sk_{AK}
$Kh_{ak}, Kh_{auth}, R, H_H, H_S$
sk_{RA}
P =DefinePolicy$(R, i \leftarrow$ Available Index$)$
TPM2_Create(P, Kh_{ak})
$(sk_{p_{ak}}; pk_p)$
$sk_{p_{ak}}, pk_p$
pk_p, H_{H_i}, H_{S_i}
$\sigma_{PoR} = Sign(pk_p, sk_{ra})$
σ_{PoR}
$\{H_{H_i}, H_{S_i}\}$ =Locate(pk_p)
$H_x \in \{H_{H_i}, H_{S_i}\}$
$\sigma_r = Sign(H_x, sk_{ra})$
σ_r, h_x
i =Locate(h_x)
TPM2_LoadExternal(pk_{ra})
$Kh_{ra} \rightarrow pk_{ra})$
kh_{ra}
TPM2_StartAuthSession()
$s \rightarrow fresh\ s$
s
SatisfyPolicy$(s, h_x, \sigma_r, Kh_{ra}, Kh_{auth})$
TPM2_NVSetBits$(b(i), R)$
$R \oplus b(i))$
$\iff R_{policy} == s$

Figure 4.7: Revocation (Registration and Revocation)

Chapter 5

Secure Information & Attestation Data Exchange in ASSURED Use Cases

After having defined all the crypto primitives and protocols provided in ASSURED, as part of the overall **Secure Information & Attestation Data Exchange** mechanism, in what follows we proceed with a detailed mapping of how these features are leveraged towards achieving the **secure data sharing requirements**, in the context of the envisioned use cases, as specified in D1.4 [15] and further elaborated in D4.1 [14]. The latter deliverable essentially translates the initially defined data sharing behaviours of all actors, in each use case environment, to the **fundamental security, privacy, integrity, trustworthiness, authentication and authorization requirements for the secure data sharing and access** that need to be achieved through the overall ASSURED Distributed Ledger & Blockchain-based Smart Contract infrastructure. Recall that data, in the context of ASSURED, can refer to **operational** (and other) information, from the deployed CPSoS, but also information (**attestation data**) from the attestation enablers that can be used as evidence of the devices' integrity and correct execution of the validation properties of interest [18]; *critical software processes to the overall operation of the target system that needs to be accompanied with strict security claims on their correct control-flow execution*.

This modeling served as the driving factor when designing all secure data management and on-chain interactions throughout the entire lifecycle of a system's operation (as described in Chapters 3 and 4): from its *registration and enrollment* to the target network, in order to create all the necessary keys and credentials for its later secure participation and operation to the *secure (and anonymous when needed) recording of attestation and threat intelligence data*, through the designed DLTs, and *sharing* to external stakeholders with the appropriate privileges as securely produced through presented Verifiable Credentials (VCs) [23]. The endmost goal is to offer all the necessary functionalities to enable **data confidentiality, integrity and multi-level access control (security by design), data ownership safeguarding (privacy by design) and traceable and credible security auditing workflow of all shared operational assurance data representing the level of trustworthiness for the entire SoS ecosystem**.

Therefore, in the following sections, we start by mapping these high-level security requirements (as defined in D1.1 [17]) to the cryptographic primitives and trust extensions offered by ASSURED (Section 5.1) for enhancing the cyber-health of next-generation "Systems-of-Systems" through securing the device communication and safeguarding the attestation data sharing over the designed Blockchain infrastructure [14]. Such high-level requirements actually depict the functional specifications of the ASSURED framework when it comes to the secure data management and, thus, we document how these functionalities are enabled by the designed protocols. Based on this detailed mapping, we then proceed (section 5.2) with describing how the data sharing behaviours of all actors, in each one of the envisioned use cases, are facilitated through the advanced ASSURED secure data management protocols, as defined in Chapters 3 and 4.

Recall that when it comes to the type of data actors encountered in ASSURED, we consider the following types [15] that participate in the secure exchange of operational and/or attestation data:

Data Subject Category	Description
Data Subject	This role reflects the identified asset (Personnel/Worker or deployed edge device/sensor) who is providing operational and attestation raw data for further processing. The transmission of the data must be done through secure and authentic channels so as to enable data integrity, authentication of the source device and authorization of the stakeholder to read and process the data.
Data Controller	The person or (software) asset (e.g., System Administrator, Public Authority, Agency, Database or other cloud-based decision support system) which alone or jointly with other can determine the purpose and means of the processing of the collected raw data. The data controller essentially defines the policies based on which specific operations are to be executed and how (level of security protection and privacy) the exchanged data is to be processed. For instance, a data controller can define the policy that needs to be executed when a new device wants to join a network; e.g., the type of system properties to be verified (in a privacy-preserving manner if dictated by the vendor of the device) prior enrolment.
Data Processor	A natural person, public authority, agency or other body party that processes operational and/or security related data on behalf of a data controller. For instance, consider the case of a Certification Body that might request access to the security attestation evidence for a given CPS environment, from the System Administrator (as the Data Controller), in order to certify/audit the correct state and operation of the entire supply chain.
Data Recipient	A natural person, public authority, agency or another body (internal or external to current network), to which the operational and security related data are disclosed, whether a third party or not. Only stakeholders with the appropriate privileges may access such sensitive information.
Third Party	A natural person, public authority, agency or body (external to the supply chain) other than the data subject, controller, processor and persons who are authorized to process personal data. Sometimes third parties can act as processors, but usually are vendors and other outside stakeholders which if performing any processing of personal data, it shall be governed by a binding contract.

Table 5.1: ASSURED Data Actors & Roles Exchanging Operational and Attestation Data

5.1 High-Level Security Requirements Linked to ASSURED Cryptographic Protocols

Table 5.2 contains the list of all relevant ASSURED technical and functional requirements that must be achieved through the integration of the defined cryptographic schemes for the secure on- and off-chain interactions with the Blockchain infrastructure, as defined in D1.1 [17] towards achieving the overall level of trustworthiness described in D2.2 [19]. Recall that the core services (as were fleshed out in D1.1 [17]) to be protected are the following:

- **Data confidentiality of the operational data** to be exchanged between the deployed devices;
- **Data integrity and verification of the attestation data** provided by a Prover device to the Verifier who in turn relays them to the Security Context Broker (SCB) for **certifiable**

recording on the ledger and **authorized and accountable sharing** to only allowed internal and/or external stakholders [24];

- **Privacy protection of the exchanged operational and/or attestation data based on the type of sensitive information included**. For the former, different levels of **privacy configuration** (e.g., full vs partial anonymity, unlinkability, untraceability, etc.) should be supported depending on the operational needs. For instance, in the "*Public Safety*" scenario where video streams with personally identifiable information are exchanged, these should be fully anonymized. For the latter, **attestation evidence privacy** should be enabled: Provers should not be required to expose configuration and execution details as part of the measurements sent to the Verifier. Attestation and verification should be achieved in a privacy-preserving manner by enabling Provers to attest to their device configuration correctness and/or a critical program's correct execution in **zero-knowledge** while enabling the entire network to benefit from the security guarantees of the ASSURED remotely verifiable attestation enablers;
- **Data confidentiality of the accompanying system/attestation raw data** (control-flow or configuration traces) as extracted during the execution of the respective attestation task;
- Release of attestation-related information, to stakeholders, with **different levels of access and information granularity**;
- **Secure design of a new device to be securely onboarded to the overall environment** by hosting a **valid and certifiable trust anchor** featuring properties such as secure key management, secure storage, and management of Verifiable Credentials (VCs) based on which self-issued verifiable presentations can be created including the verifiable evidence (from the attestation output) as attributes on their level of trustworthiness.

High Level Security Requirements	**Cryptographic primitives**
FR.AT.1 - Secure Remote Attestation Protocol	This is enabled by the underlying TPM-based Wallet supporting the **integrity of a produced attestation report** (based on **authentic traces** monitored on the Prover side) by providing the appropriate (and policy-protected) Attestation Key (AK) to be used for signing, as described in Sections 3.4 and 4.3. *The Attestation Key is securely created during the Device Enrolment phase (Section 4.1) and is binded to the correct state of the device meaning that it can only be used when the device configuration has not been altered.* The respective AK, of the Prover, is used for signing the traces (attestation raw data) sent to the Verifier who then calculates the attestation report which is finally signed (using the Verifier's AK) prior to be recorded on the ledger. Thus, integrity of both the traces and the attestation report is achieved. Finally, since the **AK of each TPM-based Wallet is unique**, this enables the **authenticity of the exchanged attestation-related data**. This is achieved by the following two features: The AK is protected under the TPM's Endorsement Key (created as part of the TPM endorsement hierarchy [13] which means that it can only be used if the host TPM has access to this specific EK) and the overall TPM Wallet operation is protected by a hardware-based key; the DAA Key [23]). This latter property enables the creation of (self-issued) Verifiable Presentations (VPs) including security claims on the authenticity of the sender device, and with verifiable guarantees that this VP belongs to the claimed entity, thus, also assuring the authenticity of the exchanged data.

FR.AT.5 - Trustworthy Prover Device Monitor (Trust Anchor)	This property pertains to the **existence and certification of a valid TPM-based Wallet**, as a trust anchor in each device, for supporting the establishment of **secure and authentic communication channels** (with other devices as well as the ASSURED backend infrastructure) and the **secure on-chain interactions** with the Blockchain infrastructure. As described in Section 4.1, during the Device Registration and Enrolment phase, the first step for the device is to attest to the Privacy CA the presence of a valid and certified TPM as the underlying root-of-trust. This is achieved by certifying the TPM's Endorsement Key credentials. The TPM, which constitutes the core building block of the ASSURED Device Wallet, thus, converting each **device to a hardware-based security token**, supports all the offered crypto primitives and schemes including **Attribute-based Encryption, Secure Communication, and (privacy-preserving) Platform Authentication**.
***FR.DLT.3* - Blockchain Hash Function**	All attestation reports are hashed by the Blockchain Peer [24] prior to their recording on the ledger by leveraging the hash calculation mechanisms described in Section 2.5. **This enables the efficient management of the data stored on the ledger and their quick query and extraction from internal entities/devices and external stakeholders with the appropriate privileges**. As also described in D2.5 [24] and D5.2 [20], this property allows the execution of such (usually resource-heavy) data operations (e.g., write and read from the ledger) by directly executing the appropriate functions as *chaincode*, thus, enabling one of the core ASSURED technical innovations revolving around the provision of **chaincode-as-a-service**. Essentially, all data management, search and sharing services (for the attestation data stored on the ledger) are implemented as chaincode (through the adoption of the gRPC technology [20]) that enables the definition and execution of such business logic to take place directly on the Blockchain network, thus, incrementally increasing the efficiency of these operations (through a set of contracts covering common terms, data, rules, concept definitions and processes).
FR.SC.1 - Secure and Efficient Cryptography	This pertains to safeguarding the data confidentiality and integrity of the different types of data (attestation and/or operational) exchanged between the deployed devices and the ASSURED backend infrastructure. It is achieved by leveraging both the **Symmetric and Asymmetric Encryption** schemes adopted in ASSURED, as described in Section 2.1. Symmetric crypto is used for establishing secure communication channels while asymmetric crypto enables the authentication and authorization of participating devices. Both features are enabled by the TPM-based Wallet that is responsible for the secure key management and the handling of Verifiable Credentials. **Attribute based Encryption** is also for protecting sensitive information (attestation raw data including control-flow and configuration traces) in order to make it available only to stakeholders that posses the necessary attributes (**different level of access and information granularity**), while **Searchable Encryption** is used for locating encrypted sensitive information, without however revealing its contents. Authentication and access to the information will be initiated firstly through the **Secure Device Enrollment**, and then through **Attribute Based Access Control** conducted he Security Context Broker (SCB).
FR.SC.2 - Flexible and Reliable Key Management	Key Management will be done through the TPM-based Wallet [23] leveraging the properties of the TPM, as a trusted component described in Section 3.1, when it comes to the **secure key creation and storage**, and **key usage protection based on policies established during the Device Registration and Enrolment phase** (Section 4.1). As depicted in Figures 4.1 and 4.2, when a device requests to be enrolled in the overall network, it will first be asked to attest the presence of a valid TPM (thus, achieving **provision of root keys and certificates at the time of registration**) which will then be used to create the Attestation and DAA Keys binded to a policy (circulated by the SCB) depicting the correct configuration state that the device needs to be prior to be allowed to use these restrictive signing keys. The same process can also be followed when creating **traditional RSA keys for protecting the data confidentiality of exchanged data** - such a symmetric key is also protected by a policy that states the conditions based on which the device is allowed to use the key for encrypting/decrypting exchanged data. Such sets of

	policies and their integration and use for key protection usage (by the TPM) in a remote attestation protocol has been formally verified in [27]. Overall, ASSURED policies and cryptographic protocols manage the entire lifecycle of ASSURED user keys from their generation to their destruction, describe the key credential that identifies the key attributes and usage restrictions. They also specify whether there can be a link between some of the keys that belong to the same user or group of users so that resulting signatures can be linked back to their origin (as is the case for one of the modes of operation required in swarm attestation [22]). Finally, these functionalities offered by the TPM-based Wallet also enable the **provision and management of application-specific keys and/or credentials**, as they can be securely loaded and accessed by only specific processes, based on dynamic policies. This, in turn, allows for **user-controlled (or owner-controlled) security for devices** based on the service needs when it comes to identity and authentication management (e.g., user-controlled anonymity, unlinkability and privacy-preserving platform authentication as offered by the ASSURED DAA scheme [16]).
FR.SC.3 - Access Control	Access Control will be handled by the SCB [24] which will be able to check the **(verifiable) credentials presented by a device or user** that wishes to access operational and/or attestation data or even when devices query for extracting specific attestation policies to execute. This feature is again supported by the TPM-based Wallet, of the device, which during the device registration and enrolment, it interacts with the Blockchain Certification Authority for securely receiving the issued credentials, including all necessary attributes that first have been verified by the Privacy CA, for then invoking the on-chain services that is entitled to (Figure 4.3). Through the use of the "*chaincode-as-a-service*" paradigm, Attribute-based Access Control is performed as a chaincode after receiving the verifiable presentations (from the devices) based on the issued credentials and including the list of attributes that are required for getting access to this specific service or information. The use of such Verifiable Presentations (VPs) is protected through the use of hardware-based keys (DAA Key of the TPM-based Wallet [23]) for binding the VPs to the specific wallet of the device and the set of required attributes. This, in turn, allows for **continuous authentication, authorization and access control based on such protected VPs which embody security claims as attributes**.
FR.SC.4 - User Privacy	To satisfy this requirement, the ASSURED framework will make sure that it collects from the user the information that is essential for its operation and nothing more. The identity of each user/device will be actually protected by performing the attestation using the **Direct Anonymous Attestation** scheme [16]. As mentioned also in FR.SC.2 the use of DAA enables user-controlled privacy for the devices by selecting their privacy profile when exchanging attestation or operational data; i.e., user-controlled anonymity, unlinkability, untraceability, etc.
FR.SC.5 - Cryptographic Primitives Supported for Trust-Aware Service Graph Chains	This requirement is satisfied by the provision of the whole list of crytpgraphic primitives offered by ASSURED (Chapter 2), namely the existence of **Symmetric and Asymmetric Encryption** methods and those of the **Attribute Based Encryption** and **Searchable Encryption** to protect the data, the generation of hashes relevant to the data and the transactions by the **Hash Functions**, the utilisation of different (anonymous) types of signatures to seal data or authenticate which is offered by the different **Digital Signatures**, and the **Direct Anonymous Attestation** scheme for facilitating attestation keeping ensuring data privacy and anonymity
FR.SC.6 - Post Quantum	While the support for Post Quantum crypto functions exceeds the core objectives of ASSURED, as described in Section 2.7, this still remains one of the long-term research avenues to be exploited by the ASSURED consortium even after the completion of the project. The focus would be on investigating how to convert the currently defined schemes and protocols into resilient mechanisms against quantum adversary attacks. One of the first things to be investigated (as a continuation of the efforts of previously EU initiatives including the FutureTPM EU project[1]) is

[1] https://futuretpm.eu/

	the introduction of latice-based cryptographic schemes, as part of the **Digital Signatures** to be provided by the TPM as a root-of-trust, adopting novel protocols that are expected to be post quantum resistant.
FR.SC.7 - Data Integrity and Authenticity Guarantees	**Data integrity and the establishment of secure and authentic communication channels** is delivered through the management of the appropriate keys by the underlying TPM. All exchanged information (attestation-related data and operational data) are signed by the TPM-based Wallet and the created Attestation Key (AK). This restrictive signing key, as described in FR.SC.2, does not only offer data integrity and key protection usage but is also linked to the Endorsement Key of the specific TPM Wallet, thus, **enabling continuous authorization and authentication guarantees**. Depending on the type of AK (traditional RSA Key or ECC-baded DAA Key), this can also enable **privacy-preserving platform authentication** where the receiving party can authenticate that exchanged information comes from a valid and authenticated part of the network but without been able to link it back to the specific device ID. As described in Section 2.4, different type of signature schemes (Group Signatures/Direct Anoymous Attestation) have been adopted in ASSURED for covering all the required privacy features by the different scenarios.
FR.SC.8 - Data Secrecy	For keeping attestation-related data private (control-flow and/or configuration traces), in ASSURED **Attribute Based Encryption** (Section 4.2) will be used to encrypt this information in such a way that only authenticated stakeholders will be able to have access; with different levels of access and information granularity. The novel ABE scheme is completely decentralized, thus, removing the need of a central entity managing the encryption/decryption keys. All required key material is managed by the host TPM-based Wallet which is responsible for creating the necessary ABE encryption/decryption keys only if the devices exhibits the required attributes based on re-defined policies (circulated by the SCB). Furthermore, the attestation report information will be searchable using **Searchable Encryption** [21], thus allowing interested stakeholders to be made aware of the existence of a report, without however disclosing any further information, unless those are designated to read the report (e.g. posses the right attributes).

Table 5.2: Mapping Crypto-Primitives to ASSURED High-level Security Requirements

5.2 ASSURED Secure Data Management User Stories

After describing how the secure data management requirements of ASSURED are achieved through the designed protocols and schemes, we now proceed with a concrete instantiation of how such functionalities are leveraged in the context of the envisioned use cases; ***Smart Manufacturing, Smart Cities, Smart Aerospace and Smart Satellites*** [17]. Based on the data management and sharing flows, defined in D1.4 [15], in the following sections we proceed with summarizing the **user stories revolving around the need for secure communication and sharing services** - of any operational and/or attestation-related data - and showcase how these are delivered through the aforementioned ASSURED functionalities.

ASSURED use cases have security and privacy requirements that are split into mandatory requirements and desirable ones as defined in D1.1 [17]. These high-level requirements are achieved through the integration of secure and efficient cryptographic protocols. In literature, user stories are a very high-level definition of requirements, describing a feature told from the end user's perspective (i.e., who desires the new capability), usually a user or customer of the system. A user story is short, generally one-sentence containing enough information to describe a requirement, so that we can check how this is achieved through the provided framework.

A user story typically follows a simple structure:

As a <*user type*>, I want to <*user-requirement*> so that <*reason*>

In our context, generic *user types* are defined in Table 5.1 and are further instantiated for the use cases in D1.4 [15] where for each user and device type a mapping is performed to the devices and stakeholders comprising the SoS ecosystem of each application domain.

5.2.1 Smart Manufacturing Use case

The smart manufacturing demonstrator for safe "*Human Robot Interaction*" showcases a workplace environment which comprises of industrial robotic arms alongside a space where personnel can walk within this environment. The personnel carry an Ultra-Wide Band (UWB) wireless tag that transmits 3D Cartesian co-ordinates to Ultra-Wide Band anchors mounted in the workspace. The location information as well as the robotic movements are made available to the Industrial PC for further processing - whether any decision needs to be made for stopping the movement of a robotic arm so as to avoid an accident. The IoT Gateway is an edge processing unit that consumes the above-mentioned data streams and tries to predict and avoid any sort of hazard in the workspace through custom algorithms that can control the robotic arm movements. From ASSURED perspective the demonstrator will benefit from: (i) **authentication and attestation all location data** stemming from the deployed sensors as well as **assurance of no malicious code injection to the devices**, and (ii) **protection of trustworthiness** of the workplace by avoiding unauthorized code injection and software manipulation of the custom algorithms running on the IoT Gateway.

For instance, consider the scenario where a secure communication channel among devices in a manufacturing floor is tampered causing network failure and data integrity is not verified allowing a compromised device to access the overall system. This event could result not only in physical injuries (compromising collision avoidance mechanisms) but also to an increase in the downtime of the production systems (interruption in production) causing major loss. Thus, it is of parmount importance for ASSURED to provide the best cryptographic protocols that ensure the security of the entire lifecyle of the devices: from their enrolment to the IoT Gateway, to providing attestations and regulating access control to revoking compromised devices.

In this context, the following data sharing scenarios have been achieved through the ASSURED data management functionalities (Table 5.3):

ID	As a $< Role >$	I want to $< Action >$	so that $< Reason >$	Functionality	ASSURED Secure Data Management

BIBA_01	Programme Logic Controller (PLC)	interact with the other deployed devices and/or IoT Gateway through secure and authentic channels	the integrity of the worker location data is achieved as received by the respective sensors	FR.SC.1;
				FR.SC.2;
				FR.SC.5;
				FR.SC.8
					Data stemming from the PLCs attached to the robotic arms will be encrypted (using a symmetric key) and signed (using the TPM-based Wallet EK) so that confidentiality and integrity guarantees can be achieved.

BIBA_02	Industrial PC	attest the correct state of all PLCs and receive authentic data on the robotic arms with integrity guarantees	timely safety-critical decisions can be made on stop/moving a robotic arm	FR.AT.1;
				FR.AT.5;
				FR.SC.2;
				FR.SC.7
					All data to be sent to the IPC (by the deployed PLCs) will be accompanied with verifiable evidence on the correct execution of the location calculation functions (e.g., RTL and Motion Capturing software). Both types of data will be signed using the TPM-based Wallet's AK.

BIBA_03	IoT Gateway	be sure about the correct configuration state and design of a new device to be on-boarded to the manufacturing floor network	the trustworthiness of the overall service graph chain does not get compromised	FR.AT.5; FR.SC.2	Any PLC device to join a manufacturing floor, during runtime, will have to execute the ASSURED Device Registration and Enrolment protocol. This step certifies the support of a root-of-trust to the newly onboarded device and then correctly creates the necessary keys with the appropriate key usage policies.

BIBA_04	IoT Gateway or System Administrator or External Stakeholder	be able to dynamically attest the configuration and/or behavioural state of all devices' location monitoring and calculation functions	the correctness of each device in the overall service graph chain (and, thus, their produced data) can be verified and safety-critical decisions can be made	FR. DLT.3;
				FR.SC.1;
				FR.SC.2;
				FR.SC.3
					Policy-based automated property-based control-flow attestation and configuration integrity verification protocols are executed after having been securely received by the device TPM-based Wallet (from the Blockchain infrastructure).

BIBA_05	System Administrator	safeguard the data confidentiality and privacy of all attestation evidence of the loaded Real Time Location or Motion Capturing processes	only internal and external stakeholders with the appropriate attributes can have access to	FR.SC.3;
				FR.SC.5;
				FR.SC.8
					All extracted control-flow and configuration traces will be encrypted using the ABE key, produced by the TPM-based Wallet, based on the attributes that need to be exhibited by the devices/stakeholders that want to process the data (based on attribute-based policies circulated by the SCB during the device registration).

Table 5.3: ASSURED Secure Data Management Functionalities for Smart Manufacturing UC

5.2.2 Smart Cities Use Case

The smart cities demonstrator revolves around the **secure and privacy-preserving communication of monitored data** towards enhancing the **public safety** of open-spaces - *by monitoring illicit movement of people or objects as well as any illegal air-substances such as gas*. Each open-space of interest has been equipped with edge devices (e.g., CCTV cameras and gas sensors) and a Gateway Infrastructure where the decision making processes are executed as part of the operation center; e.g., face recognition system. The collected data (video-streams, gas sensor data, etc.) are shared in a network of switches and routers/gateways, stored in a cloud infrastructure and shared with the external stakeholders' ecosystem, such as first responders in case of incidents via the cloud-based backend.

In the context of this scenario, the focus is on providing the **secure management of all infrastructure and edge devices** (as part of the entire supply chain ecosystem), for enhancing the public safety of citizens, and the **reliable, privacy-preserving and secure extraction and sharing of knowledge** from various data sources (i.e., Cameras, Emergency Management Networks, Gas Detector Sensors, etc.). The core objective is to automate the interaction between various technologies and systems comprising heterogeneous SoS-enabled ecosystems, in ensuring the physical security of citizens and visitors.

More specifically, from ASSURED perspective, the core requirements of interest pertain to the **runtime security and operational assurance of all deployed devices** and the **privacy conformity of collected data streams** so as to not violate the privacy properties of any monitored people (especially, if monitored by the CCTV cameras). For the former, the focus is on the **exchange of all required security claims prior to establishing the desired level of trust for authenticating and authorizing the communication of data from the edge devices**. Such security claims will be the output of the attestation enablers execution and might differ depending on the safety-critical nature of the services that leverage such data. For instance, **all data to be used in the context of public safety, need to have the highest level of trustworthiness**. Other operational data, for example, related to environmental monitoring services might not be as safety-critical, thus, less strict security claims will be required.

For privacy-preservation, the endmost goal is to **anonymize all data coming from the deployed edge devices so as to safeguard the location privacy of the monitored users**. This can either be achieved by processing the received video streams in order to "*hide*" any personally identified information for the users (e.g., face) and/or by **anonymizing the device which produced the monitored data, thus, hiding its location**. This will not reveal the user location since the received data while authentic will not be linked to any specific device. This is rather challenging as it depicts contradicting requirements: We aim to protect both the "*system from the users*" but also the "*users from the system*" [29].

To cover such security and privacy requirements, ASSURED has designed a novel Swarm Attestation protocol [22] based on the use of group-signatures and the Direct Anonymous Attestation (DAA) scheme [16] merged together with aggregate signatures so as to offer different levels of privacy depending on the sensitivity of exchanged information. For instance, in case full anonymity of the device's ID is required then (attestation) data stemming from a device is securely signed - using the DAA Key - which hides the device identity behind an *alias* (i.e., DAA basename) as part of the group. No other entity or stakeholder can de-anonymize the device unless they are aware of the specific **link token** that connects the signature back to the TPM-based Wallet of the device with this specific DAA Key credentials. For supporting privacy preserving attestations, ASSURED again leverages DAA which is anonymous digital signature scheme that allows the trusted component attestation service to hold the privacy preserving property. That is, the Verifier entity can check that attestation reports originate from an authentic device (with a valid and certified TPM-based Wallet) but it does not learn the identity of the device. To ensure trustworthiness and integrity of shared information flows, among diverse stakeholders, all exchanged data is performed in a cetifiable and auditable manner of the Blockchain infrastructure.

In this context, the following data sharing scenarios have been achieved through the ASSURED data management functionalities (Table 5.4):

ID	As a $< Role >$	I want to $< Action >$	so that $< Reason >$	Functionality	ASSURED Secure Data Management

DAEM_01	Chief Information Officer	attest (in privacy preserving manner) the correct configuration and execution state of all devices	verifiable evidence of the correctness of the received data streams can be guaranteed	FR.AT.1;
				FR.AT.5;
				FR.SC.2;
				FR.SC.4;
				FR.SC.5
					Policy-based automated control flow attestation and configuration integrity verification executed after having been securely received by the device TPM-based Wallet (from the Blockchain infrastructure).

DAEM_02	Chief Information Officer	enable policy-based automation for the privacy conformity of the received data streams	depending on the sensitivity of any personal identifiable information included, to be anonymized where needed	FR.AT.5;
				FR.SC.2;
				FR.SC.4;
				FR.SC.7
					Provision of user-controlled privacy for the devices. All operational- and attestation-related data are signed under the DAA scheme using different short-term anonymous credentials (i.e., pseudonyms) self-issued by the TPM-based Wallet.

DAEM_03	Internal Operator	setup secure and authentic communication channels with all deployed devices	all received data have verifiable confidentiality and integrity guarantees	FR.SC.1;
				FR.SC.2;
				FR.SC.7;
				FR.SC.8
					All data stemming from the edge devices are signed under the restrictive Attestation signing Key (managed and protected by the TPM-based Wallet) and then encrypted using a securely created symmetric key. This guarantees both the integrity and confidentiality of exchanged data. In case privacy is also required, the devices can sign all data under its DAA Key.

DAEM_04	Chief Information Officer	share all monitored data, from the deployed devices, to only those stakeholders with the appropriate privileges	confidentiality, integrity and privacy of the data is not violated	FR.AT.5;
				FR.DLT.3;
				FR.SC.1;
				FR.SC.3
					All extracted data is shared through the ASSURED Blockchain infrastructure and are safeguarded by appropriate Attribute-based Access Control mechanisms offered by the SCB. Essentially, only stakeholders that can exhibit the correct attributes through Verifiable Presentations (based on verifiable credentials that were issued and ratified during their registration and enrolment) can access the recorded data.

DAEM_05	External User	make sure that monitored data (with possibly identifiable information) will not be accessible to anyone other than authenticated public administrative roles (e.g., police officers)	my security and privacy are not in risk	FR.SC.1;
				FR.SC.3;
				FR.SC.8
					Data extracted from an edge device is encrypted using the ABE Key (of the TPM-based Wallet) prior to its recording on the ledger. Querying devices, entities or users will be able to successfully decrypt the data only if the attached TPM- based Wallet can verify the correctness of their local attributes, as defined by the respective data processing policies.

Table 5.4: ASSURED Secure Data Management Functionalities for Smart Cities UC

5.2.3 Smart & Secure Aerospace Use Case

The secure aerospace demonstrator is focused on the ecosystem designed around the aircraft. A complex system composed of several on-board services, available to both the on-board personnel and the ground engineers, communicating with the Ground Station Server (GSS). The focal point in such an ecosystem is the Secure Server Router (SSR) which is essentially the main heads programming unit in the aircraft acting as the communication "*bridge*" between all smaller Electronic Control Units (ECUs), controlling specific functionalities (e.g., landing gear, wing flaps, etc.), and the operation centre as part of the GSS. Thus, it is of paramount importance to be able to monitor and assess the trustworthiness level of the SSR and have strong security guarantees on its correct configuration and execution behavioural profile. Furthermore, since the SSR also acts as the data aggregation point for forwarding all data of interest coming from the ECUs, adequate protection measures need to be leveraged for the establishment of secure and authentic communication channels between the ECUS and SSR as well between the SSR and GSS.

More specifically, the main operations related to the SSR with respect to the ASSURED framework are: (i) **secure data collection from the aircraft sensors** (ECUs) while flying and the following **secure and authentic transmission** to a GSS when on the ground with adequate security claims on the correctness of the extracted data - through the periodic execution of the decentralized attestation enablers with the sensors as the Provers and the SSR as the Verifier, (ii) **remote authenticated maintenance and firmware updates**, (iii) **policy-based attestation** on ECU devices on an airplane, and (iv) **data storage and sharing with only authenticate stakeholders exhibiting the appropriate privileges** via the ASSURED DLT.

The core service of interest in such an environment, that also drives all the data security and assurance requirements, lies in the possibility of performing **remote software maintenance** to all the aircraft devices through the SSR. This is something that in the current aerospace industry is not supported: As a matter of fact, *any time an update or maintenance is required on any aircraft device, an entrusted and authenticated engineer has to physically go on the airplane and perform what is needed on site*. This, unfortunately, is one of the main hurdles that increases the operational cost of all commercial airline companies; hence, the endmost goal of migrating such processes in the digital world with remote maintenance capabilities. Of course, considering the safety-critical nature of any remote operation to be performed, there needs to be strong security claims on the trustworthiness of all involved actors and assets: **from the update (software asset) itself (to be certifiable and auditable that is free of any vulnerabilities or loopholes)**

to the correct state of a device both prior and after an update takes place so as to make that the required level of trustworthiness still holds after the maintenance operation.

For instance, all the device configuration and execution log traces should be monitored efficiently and securely transmitted to the GSS in order to correctly perform the verification of all received maintenance data (accompanied with attestation-related data). Selected critical local device log traces related to system compatibility and maintenance should be encrypted and available to the user after authentication. All security attestation reports and monitored traces should be stored, in a secure manner, so as requesting entities with the necessary privileges can access them correctly. Access control policies to the log files can be enforced by the system administrator.

In this context, the following data sharing scenarios have been achieved through the ASSURED data management functionalities (Table 5.5):

ID	As a $< Role >$	I want to $< Action >$	so that $< Reason >$	Functionality	ASSURED Secure Data Management

UTRC_01	System Administrator	be able to securely initiate a software or firmware update process for the SSR and all ECUs of the aircraft	each aircraft control unit is loaded with the necessary libraries and software without inherent vulnerabilities	FR.AT.1;
				FR.AT.5;
				FR.DLT.3;
				FR.SC.2;
				FR.SC.7
					SSR is responsible for verifying the correct execution of the update process while guaranteeing the secure communication of all exchanged attestation-related data (through the TPM-based Wallet): Verify the correct state of an ECU before and after the software update.

UTRC_02	Firmware Server	execute a firmware/software update process with only authenticated devices at the required state	only devices with the prerequisite loaded image can securely execute the update	FR.AT.1;
				FR.AT.5;
				FR.SC.2;
				FR.SC.5
					Only ECU devices with specific attributes are allowed to download and execute the certifiable software update from the ledger. This will be achieved through the use of Verifiable Presentations that can be self-issued by the TPM-based Wallet to be then checked by the SCB prior granting read privileges.

UTRC_03	Secure Server Router (SSR) and/or ECU	execute a firmware/software update process only with certifiable updates	the update process will not weaken the trustworthiness level of the device	FR.AT.5;
				FR.DLT.3;
				FR.SC.3;
				FR.SC.7
					All aircraft ECU devices need to be loaded with the latest version of secure software and firmware binaries and libraries. Thus, through the SSR, each ECU needs to be able to connect to the DLT infrastructure for getting the certifiable updates. All updates prior to be recorded on the ledger they will be signed using the private key of the GSS that can then be verified by the device TPM-based Wallet.

UTRC_04	Secure Server Router (SSR)	attest the correct configuration and execution state of all OEMs of an aircraft	secure remote maintenance can be offered to those (possibly) compromised devices	FR.AT.1;
				FR.AT.5;
				FR.SC.2FR.SC.7;
				FR.SC.8
					The SSR should be able to efficiently and securely attest the same system property of all ECU devices. This is achieved by the offered Swarm Attestation scheme [22] based on the use of group signatures integrated with the DAA scheme and aggregate signatures to be performed by the SSR prior to the recording of the overall result on the ledger.

UTRC_05	Communicator	setup a secure and authentic communication channel with a flying aircraft	operational data (e.g., state of the aircraft, trajectory mapping, etc.) can be exchanged with confidentiality and integrity guarantees	FR.SC.1;
				FR.SC.2FR.SC.7;
				FR.SC.8
					All data to be exchanged between the ECUS⟶SSR⟶GSS will be first signed using each device's TPM EK and then encrypted using a newly created symmetric key managed by the TPM-based Wallet. New symmetric encryption keys will be established per communication session.

UTRC_06	Ground Station	attest the correct state of each ECU throughout the entire trip of an aircraft	information on any malfunctioning or compromised OEM sensor can be securely extracted	FR.AT.1;
				FR.AT.5;
				FR.SC.2;
				FR.SC.5;
				FR.SC.7
					All attestation-related data, performed for all ECUs during a flight trip, will be securely stored in the SSR (acting as the Verifier) so that it can be then shared with the GSS upon landing. The use of aggregate signatures (as part of the swarm attestation process) guarantees the efficiency and integrity of all stored data.

UTRC_07	System Administrator	encrypt all information related to the actual level of trustworthiness for each aircraft	only external stakeholders with the appropriate privileges can certify and audit the correct operation of an aircraft	FR.AT.5;
				FR.SC.2;
				FR.SC.3;
				FR.SC.7;
				FR.SC.8
					All attestation-related data, prior to be shared by the SSR with the GSS, will be encrypted using the ABE Key of the SSR TPM-based Wallet. Encryption will be done based on keys that will be created by the TPM based on specific attributes that need to be exhibited by those stakeholders wishing the process the data - based on policies initially defined by the SCB during the SSR registration and enrolment process.

Table 5.5: ASSURED Secure Data Management Functionalities for Secure Aerospace UC

5.2.4 Smart & Secure Satellites Use Case

The smart satellites demonstrator focuses on the establishment of **secure and authentic communication channels** so that safety-critical mission updates can be securely exchanged from the Ground Station to low-earth orbit satellites. More specifically, the ecosystem comprises CubeSats operating and cooperating to execute specific mission(s) and the Ground Station, which monitors, maintains, and controls their operation. Given the communication among them, there is a need for ASSURED to confirm the **integrity of all modules cooperating to execute mission critical functions, enhance confidentiality and integrity and provide resilience of the software components (OS and Software modules) against specific attacks**. This collaboration towards the calculation and execution of a number of activities necessitates both **continuous authentication and authorization** between the devices (CubeSat and the Ground Station) but also the **dynamic trust assessment** so as to monitor the actual **level of trustworthiness** of each deployed CubeSat. This, in turn, translates to the need for each CubeSat been capable of providing strong security clains on the correct operation of all its safety-critical software modules upon request from the Ground Station.

For instance, a CubeSat operator controls the execution of Mission Application (coordinating various services) to perform **on-board satellite actions** (e.g., triggering a mission application which orients an imaging device to the requested coordinates and takes a picture). This data flow can be seen as a self-circling data flow within CubeSat and needs to be executed in a **secure environment providing verifiable evidence** back to the Operator on the correct execution of the mission application service. Furthermore, the **communication of the actual mission profile needs to be done in a secure and verifiable manner**: Only authentic and authorized CubeSat devices should be able to receive and respond to such a request. Thus, different symmetric keys for the secure transmission, reception and processing of all communicated needs to be established that can provide confidentiality and integrity guarantees.

The core property of interest in such an environment is to address the **convergence of security and safety** by enabling the establishment of the necessary trust relationships between all participating entities for cooperatively executing safety-critical functions. Considering also the **time sensitivity** of such systems since they need to satisfy real-time constraints to complete crucial tasks. Thus, it is of paramount importance to be able to balance these two dimensions: **Showcase how advanced security mechanisms that can offer the necessary security and assurance claims in the strict timing requirements and without affecting the CubeSat device's runtime behaviour**.

In this context, the following data sharing scenarios have been achieved through the ASSURED data management functionalities (Table 5.6):

ID	As a $< Role >$	I want to $< Action >$	so that $< Reason >$	Functionality	ASSURED Secure Data Management

SPH_01	Ground Station	attest the provision of root keys and credentials during the bootup of a satellite	device centric identity management can be performed during runtime when uploading new mission payloads to the satellite	FR.AT.5;
				FR.SC.2;
				FR.SC.7
					During bootup of each CubeSat, the appropriate keys need to be created by the host TPM-based Wallet to be certified with the Ground Station (acting as the Privacy CA). Thus, certifying the EK of the TPM; creating the AK based on the pre-defined key usage policies (binding key to a

					specific system state), and also the issuance of a Verifiable Credential to be used for subsequence communication authentication.

SPH_02	Ground Station	attest the correct configuration and execution state of a satellite during runtime	mission critical updates can only be communicated to non-compromised satellites	FR.AT.1;
				FR.AT.5;
				FR.SC.1;
				FR.SC.2;
				FR.SC.5;
				FR.SC.7;
				FR.SC.8
					This is achieved through the triggering of the attestation enablers by the Ground Station prior to the commencement of a mission critical update. The TPM-based Wallet of the CubeSat will extract the system measurements (from the on-boarded Tracer) and will sign them using the AK.

SPH_03	Ground Station	setup a secure and authentic communication channel with a satellite	for exchanging mission critical updates with confidentiality and integrity guarantees	FR.SC.1;
				FR.SC.2;
				FR.SC.5;
				FR.SC.8
					All mission critical updates and attestation-related data will be encrypted using a newly created symmetric key provided, by the TPM-based Wallet, based on ephemeral keys created using the Diffie-Hellman key agreement protocol [23]. The appropriate identity credential will also be attached to the exchanged attestation-related data.

SPH_04	CubeSat	interact with the loaded KUB-OS so that it can sign all mission-critical payloads	verifiable evidence on the identity of the host CubeSat can be provided (should not be feasible to impersonate the KUB-OS)	FR.AT.5; FR.SC.1	This functionality essentially depicts the need for continuous authentication and authorization between the CubeSat and the Ground Station. It is delivered by the TPM-based Wallet which manages Verifiable Presentations based on already issued credentials protected through a hardware-based DAA Key as described in D4.5 [23]. Essentially, the TPM-based Wallet will receive payloads from the KUB-OS that if authentic will then be signed and the signatures will be added as an attribute in a self-issued VP.

SPH_05	System Administrator	enable policy-based automation for authentication and key generation depending on the safety-critical level of the mission	different keys (securely stored) can be used for different mission payloads	FR.AT.5;
				FR.SC.2;
				FR.SC.8
					All exchanged data mission critical updates between the Ground Station and the CubeSat need to be protected with strict integrity and confidentiality guarantees. This is achieved by establishing ephemeral keys by leveraging the Diffie-Hellman Key Agreement protocol.

SPH_06	Consultant for Safety Assessment	securely have access to the history of the state of all deployed CubSats	their correct operation can be verified and audited for their entire lifecycle	FR.DLT.3;
				FR.SC.3;
				FR.SC.7
					All attestation-related data and system traces, based on which the verification process was executed, are signed using the ABE Key of the CubeSat so that when sent to the Ground Station (in bundles when such a communication is allowed) will be shared on the ledger with only those stakeholders that can exhibit the necessary attributes for accessing them.

Table 5.6: ASSURED Secure Data Management Functionalities for Secure Satellites UC

Chapter 6

Conclusion

This section concludes the deliverable and summarizes its findings. The scope of this deliverable is to present ASSURED cryptographic primitives with their sequence diagrams and investigate how to integrate such advanced cryptographic primitives including symmetric encryption, attribute-based encryption, hash message authentication code and digital signatures in ASSURED framework for guaranteeing the compliance to the data confidentiality, data integrity and data traceability policies defined in D3.2 [16]. D4.2 constitutes a bridge that maps each of the security and privacy requirements identified in D1.1 [17] to the associated cryptographic primitive that offers the required type and level of security and privacy needed for each of the ASSURED use cases.

In summary, deliverable D4.2 specifies the following ASSURED cryptographic protocols:

- **Secure Enrollment**: To support secure enrollment in the ASSURED Blockchain services, Deliverable 4.2 presents a secure three-phase enrollment protocol that explains the concrete issuing of the Attestation and the Blockchain key credentials as well as defining user attributes. User attributes are issued during the Secure Enrolment phase, by the Privacy Certification Authority (Privacy CA), which then credits the device a credential that is forwarded to the SCB for verification. If successful, the SCB will notify the Blockchain Certification Authority (Blockchain CA) which, in turn, will create the appropriate certificate for the device including all verified attributes based on which it can access specific attestation results. These attributes may also be bonded to the key management, for instance in the proposed Enrollment protocol, an edge device is able to create attestations using its Attestation Key (AK) only if the device is configured correctly with an authorized tracer.

- **Control Flow Attestation (CFA)**: CFA is a mechanism that validates a specific program control flow execution. To support CFA, the tracer tracks the control flows of a given program and generates a trace report of the latest control flow graph of the program. Deliverable 4.2 presents a concrete cryptographic protocol that supports the Control Flow Attestation describing the TPM-Tracer internal communication protocol for signing authorized traces.

- **Key-Policy Attribute-based Encryption (KP-ABE)**: After a Blockchain user being authenticated to a ledger, it can only have decryption rights to the encrypted data stored on the ledger if the user possesses some attributes that make correct decryption. In this Deliverable, we developed a novel Key-Policy Attribute-based Encryption (KP-ABE) that will be adopted by ASSURED to enable the device to perform data encryption/decryption using its TPM wallet.

- **Revocation**: This deliverable presents a concrete revocation scheme that enables an edge device to deactivate any credentials and short-term signature keys that were created during their enrollment phase in the ASSURED ecosystem, hence a wrongdoer can be evicted and be prevented from further participation.

After having defined all the cryptographic primitives and protocols provided in ASSURED, as part of the overall Secure Information & Attestation Data Exchange mechanism, a detailed mapping of how these features are leveraged towards achieving the secure data sharing requirements, in the context of the envisioned use cases, as specified in D1.4 [15] and further elaborated in D4.1 [14]. Finally, we want to highlight that the implementation of the designed crytographic protocols with the integration of ASSURED technical components will take place under WP5 and WP6 of the project.

List of Abbreviations

Abbreviation	Translation
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
Vf	Virtual Function
VM	Virtual Machine
Vrf	Verifier
WP	Work Package
ZTP	Zero Touch Provisioning

References

[1] Man Ho Au, Joseph K Liu, Willy Susilo, and Tsz Hon Yuen. Certificate based (linkable) ring signature. In *International Conference on Information Security Practice and Experience*, pages 79–92. Springer, 2007.

[2] Carsten Baum, Ivan Damgård, Vadim Lyubashevsky, Sabine Oechsner, and Chris Peikert. Efficient commitments and zero-knowledge protocols from ring-sis with applications to lattice-based threshold cryptosystems. Cryptology ePrint Archive, Report 2016/997, 2016. https://eprint.iacr.org/2016/997.

[3] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In *Cryptographers' Track at the RSA Conference*, pages 136–153. Springer, 2005.

[4] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. Foundations of fully dynamic group signatures. *Journal of Cryptology*, 33(4):1822–1870, 2020.

[5] Ernie Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In *Proceedings of the 11th ACM Conference on Computer and Communications Security*, CCS '04, pages 132–145, New York, NY, USA, 2004. ACM.

[6] Ernie Brickell, Liqun Chen, and Jiangtao Li. A new direct anonymous attestation scheme from bilinear maps. In *International Conference on Trusted Computing*, pages 166–178. Springer, 2008.

[7] 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, 2009.

[8] Ernie Brickell and Jiangtao Li. A pairing-based daa scheme further reducing tpm resources. In *Proceedings of the 3rd International Conference on Trust and Trustworthy Computing*, TRUST'10, pages 181–195, Berlin, Heidelberg, 2010. Springer-Verlag.

[9] Jan Camenisch. Efficient and generalized group signatures. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 465–479. Springer, 1997.

[10] David Chaum and Eugène Van Heyst. Group signatures. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 257–265. Springer, 1991.

[11] Liqun Chen, Nada El Kassem, Anja Lehmann, and Vadim Lyubashevsky. A framework for efficient lattice-based daa. In *Proceedings of the 1st ACM Workshop on Workshop on Cyber-Security Arms Race*, pages 23–34, 2019.

[12] Liqun Chen and Jiangtao Li. Flexible and scalable digital signatures in tpm 2.0. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer &#38; Communications Security*, CCS '13, pages 37–48, New York, NY, USA, 2013. ACM.

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

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

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

[16] The ASSURED Consortium. Assured layered attestation and runtime verification enablers design & implementation. Deliverable D3.2, November 2021.

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

[18] The ASSURED Consortium. Operational sos process models & specification properties. Deliverable D1.3, 2021.

[19] The ASSURED Consortium. Policy modelling & cybersecurity, privacy and trust constraints. Deliverable D2.2, November 2021.

[20] The ASSURED Consortium. Assured blockchain and data storage environment. Deliverable D5.2, February 2022.

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

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

[23] The ASSURED Consortium. Assured tc-based functionalities. Deliverable D4.5, February 2022.

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

[25] Rafaël del Pino, Vadim Lyubashevsky, and Gregor Seiler. Lattice-based group signatures and zero-knowledge proofs of automorphism stability. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 574–591. ACM, 2018.

[26] Nada El Kassem, Liqun Chen, Rachid El Bansarkhani, Ali El Kaafarani, Jan Camenisch, Patrick Hough, Paulo Martins, and Leonel Sousa. More efficient, provably-secure direct anonymous attestation from lattices. *Future Generation Computer Systems*, 99:425–458, 2019.

[27] Georgios Fotiadis, José Moreira, Thanassis Giannetsos, Liqun Chen, Peter B. Rønne, Mark D. Ryan, and Peter Y. A. Ryan. Root-of-trust abstractions for symbolic analysis: Application to attestation protocols. In Rodrigo Roman and Jianying Zhou, editors, *Security and Trust Management*, pages 163–184, Cham, 2021. Springer International Publishing.

[28] Eiichiro Fujisaki and Koutarou Suzuki. Traceable ring signature. In *International Workshop on Public Key Cryptography*, pages 181–200. Springer, 2007.

[29] Stylianos Gisdakis, Thanassis Giannetsos, and Panos Papadimitratos. Shield: A data verification framework for participatory sensing systems. In *Proceedings of the 8th ACM Conference on Security Privacy in Wireless and Mobile Networks*, WiSec '15, New York, NY, USA, 2015. Association for Computing Machinery.

[30] Shafi Goldwasser, Silvio Micali, and Ronald L Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on computing*, 17(2):281–308, 1988.

[31] International Organization for Standardization. IISO/IEC 11770-1:2010 Information technology — Security techniques — Key management — Part 1: Framework, 2010.

[32] Fabien Laguillaumie, Adeline Langlois, Benoît Libert, and Damien Stehlé. Lattice-based group signatures with logarithmic signature size. In *International conference on the theory and application of cryptology and information security*, pages 41–61. Springer, 2013.

[33] 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.

[34] 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.

[35] Benoît Libert, San Ling, Fabrice Mouhartem, Khoa Nguyen, and Huaxiong Wang. Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016*, pages 373–403, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[36] Benoît Libert, San Ling, Khoa Nguyen, and Huaxiong Wang. Zero-knowledge arguments for lattice-based accumulators: logarithmic-size ring signatures and group signatures without trapdoors. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 1–31. Springer, 2016.

[37] San Ling, Khoa Nguyen, and Huaxiong Wang. Group signatures from lattices: simpler, tighter, shorter, ring-based. In *IACR International Workshop on Public Key Cryptography*, pages 427–449. Springer, 2015.

[38] Vadim Lyubashevsky. *Towards practical lattice-based cryptography*. University of California, San Diego, 2008.

[39] National Institute of Standards and Technology. Post-quantum cryptography standardization, 1 2017. https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization.

[40] Ronald L Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 552–565. Springer, 2001.

[41] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer, 2005.

[42] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. on Computing*, pages 1484–1509, 1997.

[43] 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.