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

D4.1 ASSURED BLOCKCHAIN ARCHITECTURE

Revision: v.1.0

Work package	WP 4
Task	Task 4.1
Due date	31/11/2021
Submission date	23/12/201
Deliverable lead	SUITE5
Version	1.00
Editors	Sotiris Koussouris (SUITE5), Stefanos Venios (SUITE5)
Reviewers	Liqun Chen (SURREY), Richard Mitev, David Koisser (TUDA)
Abstract	D4.1 aims to explore the different needs of the ASSURED ecosystem related to the policy compliant blockchain infrastructure, scrutinise the most relevant and appropriate technologies that exist today which can accommodate the specific needs of the project and select the most appropriate, setting the technological foundations. Based on this analysis, this deliverable also documents the high-level conceptual architecture of the DLT Architecture, adopted in ASSURED, that serves as the cornerstone for both secure and privacy-preserving data sharing operations as well as the enforcement of the attestation policies as extracted by the Policy Recommendation Engine.
Keywords	DLT, blockchain, Architecture, Smart Contracts

Funded by the Horizon 2020
Framework Programme of the European Union

Document Revision History

Version	Date	Description of change	List of contributors
v0.1	3.09.2021	ToC	Sotiris Koussouris (SUITE5)
v0.2	17.09.2021	State-of-the-art analysis on the relevant blockchain technologies and on- and off-chain security schemes (Chapter 3) and first version of the overall ASSURED blockchain architecture (Chapter 6)	Edlira Dushku (DTU) Kaitai Liang (TUDE)
v0.3	24.09.2021	Extraction of specific functionalities needed to be supported by the ASSURED blockchain architecture based on the data sharing behaviours defined in D1.4 (Chapter 4)	Edlira Dushku (DTU) Richard Mitev, Philip Rieger (TUDA) Liqun Chen, Nada El Kassem (SURREY) Thanassis Giannetsos, Dimitris Papamartzivanos (UBITECH) Ioannis Avramidis (INTRA) Ilias Aliferis (UNIS) Sotiris Koussouris, Stefanos Venios, Alexandros Tsaloukidis, Konstantinos Charalambous (SUITE5)
v0.4	15.10.2021	Finalization of the flow of actions and mode of operation for the type of smart contracts used in ASSURED for supporting the enforcement of attestation policies (Chapter 4)	Kaitai Liang (TUDE) Liqun Chen, Nada El Kassem (SURREY) Sotiris Koussouris, Alexandros Tsaloukidis, Konstantinos Charalambous (SUITE5) Thanassis Giannetsos, George Misiakoulis (UBITECH)
v0.5	29.10.2021	Description of the Attribute-based Encryption (ABE) scheme to be supported by the TPM-based Wallet of each device (Chapter 6)	Liqun Chen, Nada El Kassem (SURREY)
v0.6	12.11.2021	Finalization of the ASSURED blockchain architecture and the functionalities of all internal components (Chapter 6)	Kaitai Liang (TUDE) Liqun Chen, Nada El Kassem (SURREY) Sotiris Koussouris, Stefanos Venios, Alexandros Tsaloukidis, Konstantinos Charalambous (SUITE5) Ilias Aliferis (UNIS) Thanassis Giannetsos, George Misiakoulis (UBITECH)
v0.7	29.11.2021	Finalization of the description of sequence of actions per ASSURED blockchain-functionality and construction of the respective diagrams (Chapter 7)	Kaitai Liang (TUDE) Liqun Chen, Nada El Kassem (SURREY) Sotiris Koussouris, Stefanos Venios, Alexandros Tsaloukidis, Konstantinos Charalambous (SUITE5) Thanassis Giannetsos, George Misiakoulis (UBITECH)
v0.8	10.12.2021	Refinements on the positioning and use of the smart contracts in relation to the overall ASSURED Architecture	Thanassis Giannetsos (UBITECH), Richard Mitev, Philip Rieger (TUDA)
v0.9	17.12.2021	Review the document	Liqun Chen (SURREY) Richard Mitev, David Koisser (TUDA)
v1.0	23.12.2021	Finalisation of the document	Sotiris Koussouris (SUITE5)

Funded by the Horizon 2020 Framework Programme of the European Union

Editors

Sotiris Koussouris (SUITE5), Stefanos Venios (SUITE5)

Contributors (ordered according to beneficiary numbers)

Edlira Dushku (DTU)

Richard Mitev (TUDA)

Liqun Chen, Nada El Kassem (SURREY)

Kaitai Liang (TUDE)

Thanassis Giannetsos, Dimitris Papamartzivanos, George Misiakoulis (UBITECH)

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

Ilias Aliferis (UNIS)

Ioannis Avramidis (INTRA)

Funded by the Horizon 2020
Framework Programme of the European Union

DISCLAIMER

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)

Funded by the Horizon 2020 Framework Programme of the European Union

EXECUTIVE SUMMARY

This deliverable lingers over the **attestation related policies and data management using smart contracts and DLTs in ASSURED**, with an in-depth analysis of SOTA approaches and focusing on the secure on- and off-chain data interactions and operations related to the secure data sharing. *Recall that one of the main innovations of ASSURED is the enforcement of all attestation policies through the use of smart contracts*.

Once the **scheduling (attestation) policies** have been compiled, as an output of the Policy Recommendation Engine, the last step of the overall security process is the deployment of this set of targeted policies over the SoS-enabled ecosystem for managing the identified risks, as well as to handle the real-time supervision and monitoring of the correct execution of these policies. This is done through the **ASSURED Security Context Broker (SCB) via the use of smart contracts leveraging the designed policy-compliant blockchain infrastructure** described in this deliverable. The SCB, acting as the *trusted operator* of the produced policies, will be triggered by the Policy Recommendation Engine for converting the attestation policies into smart contract logic and further deploy the smart contract to the ledger. All **attestation-related processes and data** will be recorded and kept in a **traceable and accountable manner** on the **permissioned ledger infrastructure**, thus, providing a **credible security auditing and certification** workflow.

In this context, the overall design of the ASSURED blockchain infrastructure with a detailed break-down of all internal components and operations are outlined, including the data flow envisioned within ASSURED between participating entities. *This sets the scene for the concrete functionalities and algorithms of ASSURED enhanced data security, integrity and privacy mechanisms as outlined in Chapters 6 and 7*.

In this direction, ASSURED enables enhanced **data security, integrity privacy and ownership safeguarding** (security- and privacy-by-design) and **data provenance and sovereignty** checking mechanisms. The platform uses **private and public** Blockchain-based distributed ledgers for offering enhanced data and transaction security. To this end, ASSURED protects operational and attestation-related data and resources against leak or improper modifications, while at the same time ensures data availability to legitimate users and devices. Internal storage and ledger infrastructures, handling attestation related data and monitored system traces, can track its provenance and are regularly audited to comply with specified security and privacy policies and regulations. This way devices are in control of their own privacy and that of their data. For the former, privacy requirements are described through privacy-related policies where the type of crypto primitives to be used by edge devices are described towards achieving the necessary properties of *anonymity, unlinkability, unlinkability and unobservability*. These policies are afterwards translated in the appropriate smart contracts, following the principle of user privacy empowerment. Depending on the required privacy level, privacy enhancement is achieved through the use of the TPM-based Wallets as a central building block towards the provision of privacy-preserving signature schemes (Direct Anonymous Attestation (DAA)). By assuring auditable, security and privacy policy compliant actions, ASSURED also guarantees that application ecosystems where such policies have been technically enforced are highlighted.

ASSURED designed crypto primitives for supporting the necessary secure on- and off-chain interactions rely on: (1) **Attribute-based Access Control** for verifying that devices (authenticated members of the overall SoS-enabled ecosystem) and external stakeholders (users that are not part of the target ecosystem but wish to get access to a sub-set of attestation results for certifying the overall state of the SoS-enabled ecosystem or checking the level of assurance and trustworthiness of specific devices and processes), requesting access to

Funded by the Horizon 2020 Framework Programme of the European Union

specific attestation related data, have the correct privileges and attributes, (ii) **Attribute-based Encryption** for guaranteeing the **confidentiality of the accompanying attestation data** (system traces) – *ASSURED proposes the use of a novel, decentralized attribute-based encryption scheme, thus, enabling the overarching theme of the project towards shifting trust from the infrastructure to the edge*, and (iii) **Searchable Encryption** for external stakeholders been able to perform queries in an efficient manner.

All these primitives are enabled by the Trusted Blockchain Wallets, designed in ASSURED, used to: (i) provide strong **user authentication** and to **securely store the user credentials based on the TPM's secure key storage**, (ii) **control and authorize access to private or public ledger channels** based on the user authentication process (e.g., to authorize access to or operations on different ledgers), and (iii) **securely and efficiently verify blockchain updates**. In this way, ASSURED will significantly advance the state-of-the-art of blockchain verification methods

The overall purpose of this deliverable is to provide a reference document on the security and privacy-preserving trust anchors that have been selected by the consortium for integration in the overall ASSURED platform towards achieving one of the main visions of secure attestation policy deployment, execution, recording and sharing through the use of policy-compliant blockchain structures. This will be used as input to the implementation of the platform's architecture, the functionality of platform's security sub-components and the further investigation, design and development of the core ASSURED security, privacy and trust bundles.

Funded by the Horizon 2020
Framework Programme of the European Union

TABLE OF CONTENTS

1 INTRODUCTION 11
1.1 Introduction and Scope 11
1.2 Relation to other WPs and Deliverables 11
1.3 Deliverable Structure 13
2 DLTS IN ASSURED - WHY IS THERE A NEED? 15
2.1 Role of DLTs in ASSURED 15
2.1.1 Need for Having DLTs for Data Sharing in ASSURED 15
2.1.2 Need for Having DLTs to Facilitate ASSURED Trust Guarantees 16
2.2 Interaction with other Components 17
3 DLT AND DATA PROTECTION TECHNOLOGIES 22
3.1 Setting the Context 22
3.2 Blockchain Types and Platforms 24
3.3 Smart Contracts 26
3.4 Access Control mechanisms 27
3.5 Consensus algorithms 30
3.6 ASSURED Security and Crypto Primitives for Secure Data Management 33
3.6.1 Dynamic Searchable Symmetric Encryption 33
3.6.2 Attribute-based Encryption 34
3.7 Summary of Analysed Technologies 35
4 ASSURED BLOCKCHAIN RELEVANT REQUIREMENTS 37
4.1 DLT Sharing Requirements 37
4.2 DLT Security, Confidentiality and Trust Requirements 40
5 SMART CONTRACTS IN ASSURED 44
5.1 Types of Smart Contracts in ASSURED 44
5.1.1 Smart Contract Entities and their Roles 45
5.2 Smart Contracts Workflows 46
5.2.1 A General Description 46
5.2.2 Specific Setting in ASSURED 49
6 THE ASSURED BLOCKCHAIN ARCHITECTURE 53
6.1 Architecture Overview 53
6.2 Architectural Building Blocks - Private and Public Channels 57
6.2.1 DLT Blockchain Peer 57
6.2.2 Chaincode Component 58
6.2.3 Private and Public Ledger 58

Funded by the Horizon 2020 Framework Programme of the European Union

6.2.4 Searchable Encryption Component 59
6.2.5 API Layer 59
6.2.6 Ordering Service 59
6.2.7 Security Context Broker 60
6.2.8 Membership Service Provider (ABAC service) 62
6.3 Architectural Building Blocks - Device Level 62
6.3.1 TPM-based Wallet 62
6.3.2 Attribute Based Encryption (ABE) Component 63
6.4 Architectural Building Blocks - Organisation Boundaries Level 66
6.4.1 Smart Contract Composition Engine 66
6.4.2 Privacy Certificate Authority (Privacy CA) 67
6.4.3 Blockchain Certificate Authority 70
6.4.4 Off Chain Storage 71
6.4.5 Indexing Service 71
6.5 Mapping Architecture Blocks to Requirements 72
7 ASSURED DLT OPERATIONS EXPLAINED 74
7.1 Smart Contract Deployment 74
7.2 Joining/Accessing a Private/Public Channel 75
7.2.1 Providing Access to External Entities to the Public Channel 77
7.2.2 Providing Access to the Private Channel to (new) devices within an Organisation 77
7.3 Placing Attestation Results to the Ledgers 77
7.4 Querying the Ledger for Attestation Results 81
7.4.1 Private Ledger Querying 81
7.4.2 Public Ledger Querying 82
7.5 Extracting Attestation Results from the Ledgers 83
7.5.1 Extracting Attestation Evidence from the Private Ledger 83
7.5.2 Extracting Attestation Data from the Off-Chain Storage 84
8 CONCLUSIONS 86
APPENDIX A 93
A.1 A Bytecode Example 93

Funded by the Horizon 2020 Framework Programme of the European Union

LIST OF FIGURES

FIGURE 1: RELATION OF D4.1 TO OTHER WPS AND DELIVERABLES 12
FIGURE 2: ASSURED SECURITY PROCESS AND INTERACTION WITH BLOCKCHAIN INFRASTRUCTURE 18
FIGURE 3: A GENERAL CONVERSION OF SMART CONTRACT 47
FIGURE 4: A SOLIDITY EXAMPLE 47
FIGURE 5: TURN SMART CONTRACT INTO TRANSACTION 48
FIGURE 6: THE EXECUTION OF SMART CONTRACT 48
FIGURE 7: ATTESTATION BASED SMART CONTRACT DEPLOYMENT 49
FIGURE 8: ATTESTATION SMART CONTRACT CHALLENGE GENERATION 50
FIGURE 9: ATTESTATION SMART CONTRACT VERIFICATION. 51
FIGURE 10: ASSURED DLT ARCHITECTURE 56
FIGURE 11: ATTRIBUTE-BASED ACCESS CONTROL PERFORMED BY THE MSP 61
FIGURE 12: ASSURED ABE SCHEME USING THE TPM WALLET 65
FIGURE 13: THE ATTESTATION CERTIFICATE AUTHORITY SOLUTION IN TPM 2.0 (ACAS-IN-TPM-2.0) REQUEST GENERATING A “RESTRICTED" SIGNING KEY AK 69
FIGURE 14: SECURE DEVICE ENROLLMENT 69
FIGURE 15: SMART CONTRACT DEPLOYMENT 75
FIGURE 16: PUBLIC CHANNELS (LEDGER) ACCESS 76
FIGURE 17: PRIVATE CHANNELS (LEDGER) ACCESS 78
FIGURE 18: SEQUENCE DIAGRAM - PLACING ATTESTATION RESULTS TO THE LEDGERS 80
FIGURE 19: SEQUENCE DIAGRAM - PRIVATE LEDGER QUERYING 82
FIGURE 20: SEQUENCE DIAGRAM - PUBLIC LEDGER QUERYING 83
FIGURE 21: SEQUENCE DIAGRAM - EXTRACTING ATTESTATION EVIDENCE FROM THE PRIVATE LEDGER 84
FIGURE 22: SEQUENCE DIAGRAM - EXTRACTING ATTESTATION RESULTS FROM THE OFF-CHAIN STORAGE 85

Funded by the Horizon 2020 Framework Programme of the European Union

LIST OF TABLES

TABLE 1: TYPES OF BLOCKCHAINS 24
TABLE 2: MAIN BLOCKCHAIN PLATFORMS COMPARISON 26
TABLE 3: THE COMPARISON OF CONSENSUS ALGORITHMS 33
TABLE 4: SUMMARY OF TECHNIQUES 35
TABLE 5: ASSURED DLT SHA REQUIREMENTS - 39
TABLE 6: ASSURED DLT SCT REQUIREMENTS - DATA SECURITY AND INTEGRITY 40
TABLE 7: ASSURED DLT SCT REQUIREMENTS - AUTHORISATION AND ACCESS CONTROL 41
TABLE 8: ASSURED DLT SCT REQUIREMENTS - CRYPTOGRAPHY 42
TABLE 9: ASSURED DLT SCT REQUIREMENTS - DATA ENCRYPTION 42
TABLE 10: ASSURED DLT SCT REQUIREMENTS - TRUSTWORTHINESS OF EXCHANGE DATA 43
TABLE 11: ASSURED DLT SCT REQUIREMENTS - NON-REPUDIATION AND ACCOUNTABILITY OF ACTIONS 43
TABLE 12: ENTITIES RELEVANT TO ASSURED SMART CONTRACTS 45
TABLE 13: ARCHITECTURE BLOCKS MAPPED TO REQUIREMENTS 73

Funded by the Horizon 2020
Framework Programme of the European Union

1 INTRODUCTION

1.1 INTRODUCTION AND SCOPE

The present deliverable D4.1 “ASSURED Blockchain Architecture” comes as the first deliverable of WP4 of the ASSURED project which is tasked to design and implement the blockchain-based prototype tools, APIs and components that will support the blockchain infrastructure of the ASSURED project.

As such, this deliverable aims to explore the different needs of ASSURED relevant to a blockchain infrastructure, scrutinise the most relevant and appropriate technologies that exist today which can accommodate the specific needs of the project and select the most appropriate, setting the technological foundations that should be considered for the design and the implementation phases.

These activities, when combined with the high level requirements and the concepts brought forward in the earlier stages of the project, allow to flesh out more detailed requirements relevant to the DLT to be developed in ASSURED, in order to cover both the sharing functionalities envisages, and the security and trust guarantees that need to be put in place, to enable a fully transparent and accountable secure information exchange based on traceable and credible security auditing activities.

It is based on these requirements that the work in this deliverable continues to formulate in a more detailed manner how the overall operation within the ASSURED DLT should be running based on the development of smart contracts that will automate most of the transactions that need to take place, while also these requirements drive the design of the overall DLT infrastructure architecture. The latter is evidently happening by allowing engineers to better understand the role of the various components that have been included in the initial ASSURED concept, and at this point by mapping those to the requirements, the different component take flesh as key architectural elements that now have defined tasks to support and that clearly showcase how they can interact with the rest of the entities that are part of either the ASSURES DLT Architecture, or of the extended ASSURED ecosystem.

In this context, the architecture presented in this deliverable, and the initial flows and interactions of components described will act as the blueprint for the consequent tasks of this WP, towards the realisation and the design of the internal operations and the application logic relevant to the different ASSURED blockchain components defined in the architecture, that mostly have to do with the definition of the algorithms, the methods and the mechanisms to bridge the services of data sharing with those of security, privacy preservation and trust generation, always under the context of ASSURED.

1.2 RELATION TO OTHER WPS AND DELIVERABLES

As identified above, D4.1 comes as the first deliverable of WP4 “*Blockchain-based ASSURED Supply Chain Control Services and Trust Evidence Collection*” and is acting as the reference document for the design of all secure data on- and off-chain operations that are envisioned for the other tasks and deliverables of this work package, describing how the different components should be designed internally in order to integrate them and work harmonically with the overall

Funded by the Horizon 2020 Framework Programme of the European Union

concept of the ASSURED DLT network relevant data sharing and security and trust guarantees provisioning.

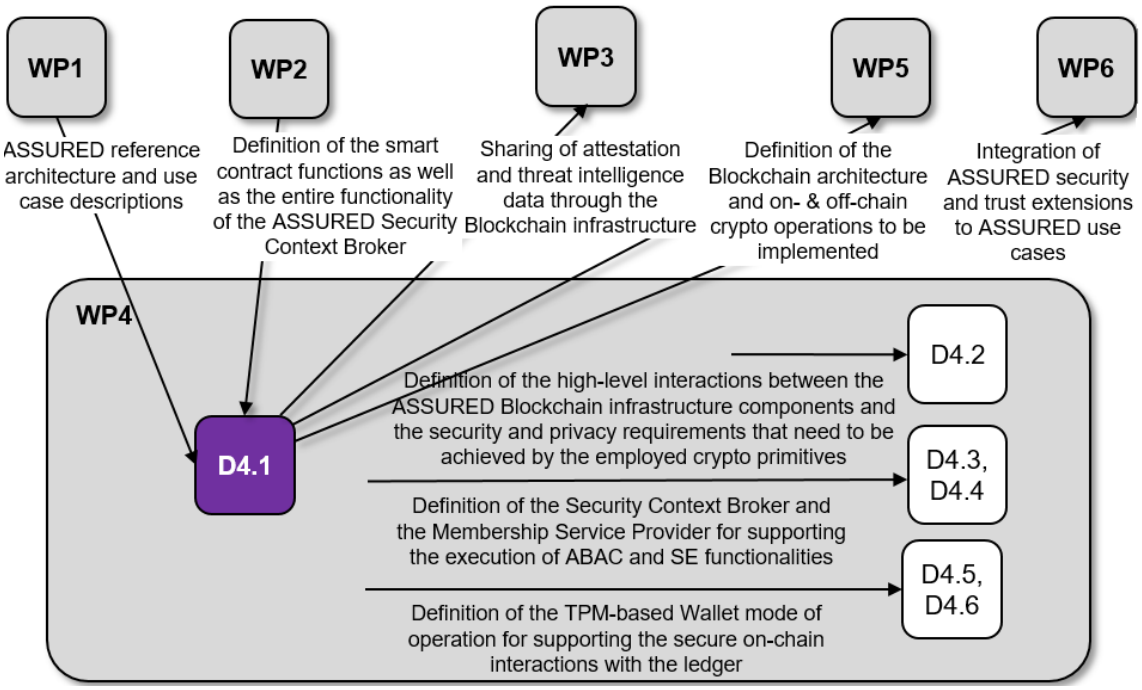


FIGURE 1: RELATION OF D4.1 TO OTHER WPS AND DELIVERABLES

As such, as depicted in FIGURE 1, D4.1 is providing input to all the other T4.x activities in order for them, as specified in the DoA, to leverage this knowledge for:

- design and implementation of related advanced, lightweight crypto algorithms that will be used to enhance **decentralised data integrity, access control and multi-layer data sharing** across the entire SoS-enabled supply chain (D4.2);
- defining the functionalities – to be supported by the Security Context Broker (SCB) – as it pertains to the **secure management of all data sharing functionalities**; either for operational or threat intelligence (attestation-related) data. This includes the design of appropriate **Searchable Encryption (SE)** and **Attribute-based Access Control (ABAC)** mechanisms for the secure and authenticated access to the deployed distributed ledgers (D4.3 and D4.4), and
- implementation of all the advanced crypto operations to be supported by the TPM-based Wallet – that is instantiated in each edge device – for the secure interaction with the ledgers: both when it comes to **querying (reading) the attestation policy**, to be executed, and for **recording the attestation report back on the ledger**. Recall that one of the core innovations of ASSURED is the enforcement of all scheduling attestation policies through smart contracts, as described in the overall ASSURED reference architecture (D4.5 and D4.6).

In this context, requirements per blockchain component are delivered in this document, and interactions between these components are defined, without however going into application-level requirements, as those will be defined and delivered during the code development phase of the project under this WP5 (see below).

Funded by the Horizon 2020 Framework Programme of the European Union

The main input for the elaboration of this deliverable comes from the work already been delivered as part of WP1, where the different components of the overall ASSURED architecture, including also the blockchain infrastructure, have been described, requirements for the different **sharing and trust services to be provided have been set at a high level** and the use cases have described their envisaged way of operating, shedding light into the path that should be taken for delivering the services for secure and trustworthy data exchange. For the latter, D1.4 [52] has already put forth the description of the **security, privacy and trustworthiness requirements that need to adhere all the data sharing behaviours for the envisioned use cases**. These requirements set the scene for defining the functionalities that need to be provided by all the blockchain-related components and services (Chapter 4).

There are also close links between this deliverable and the work performed under T2.4 "*Smart Contract Definitions for Policy Enforcement and Security Data Management*" of the project, as the **smart contract logic** (functions for supporting the attestation policy *deployment, execution and recording*) to be defined under that task has to be falling in scope with the architecture that has been developed in this document, while the architecture needs to make sure that the different components introduced are the ones that can accommodate the needs and the operating of the smart contracts, following the requirements of those contracts and the requirements of the attestation needs which are researched in D2.2 [53]. These scheduling attestation policies revolve around the execution of the attestation schemes described in D3.2 [51]. The blockchain architecture needs to be able to accommodate their secure and auditable recording and sharing: *All attestation-related data will be recorded and kept in a traceable and accountable manner on the ledger infrastructure, thus, providing a credible security auditing and certification workflow*.

Moreover, the architecture presented in this deliverable will also serve as the blueprint to be used for the implemented of the overall ASSURED DLT network that will happen in the frame WP5 "ASSURED Framework Integration", under Task 5.3 "Secure Information & Attestation Data Exchange Services Implementation", and as such deliverable D4.1 has a direct link with the implementation and integration activities of the overall ASSURED framework.

1.3 DELIVERABLE STRUCTURE

This deliverable is structured as follows.

In **Chapter 2** we first discuss the role of DLTs relevant to ASSURED, to allow the reader to grasp the main context of how DLTs support the overall operation of the framework and what are the main considerations and issues we need to tackle.

Following comes **Chapter 3**, where a high-level review of the state of the art in technologies relevant to those we are going to use in the ASSURED DLT infrastructure, covering not only core blockchain technologies, but also other aspects that have to do with security and cryptography, which in ASSURED will be embedded in the DLT infrastructure, such as Attribute Based Encryption and Searchable Encryption

Chapter 4 elaborates on the work performed under WP1 and extracts out of the different high-level concepts, data flows and use cases the requirements that are relevant to the ASSURED DLT infrastructure. These are divided into "Data Sharing" requirements and "Security, Privacy and Trust" requirements.

Funded by the Horizon 2020 Framework Programme of the European Union

Chapter 5 discusses the role of smart contracts in the scope of ASSURED, starting from discussing their main operations and elaborating on how these can be used to cover the specific needs of the ASSURED framework.

In **Chapter 6**, we present the DLT architecture of ASSURED, defining the key components that need to be developed, and provide certain details relevant to their operations. Moreover, a mapping of the different components to the requirements identified in Chapter 3 is done, to serve as the basis of the development activities of the network.

Following the design of the architecture, **Chapter 7** presents the main flows relevant to the operations that will be supported by the ASSURED DLT, showcasing the interactions between the different ingredients of the architecture presented in the previous chapter.

Finally, **Chapter 8** concludes the report and provides a summary of the presented work.

Funded by the Horizon 2020
Framework Programme of the European Union

2 DLTS IN ASSURED - WHY IS THERE A NEED?

2.1 ROLE OF DLTS IN ASSURED

Over the past years and following the advent of the data era and of globalisation, it became evident that enterprise systems should capitalise not only on the internally produced data, but also on data they can access and acquire from external sources. At the same time, the introduction of IoT and other similar infrastructures within organisations led to the exponential generation of data, which fuelled the movement of digitalisation of operations, which is based on exploiting all data that can be accessed to **extract intelligence that can be used to improve operations, not only from a performance viewpoint, but also from a security, privacy, and trust perspective.**

Most of today's trusted systems, such as banking and e-mail services, use highly centralised approaches to security. Such trusted systems rely on trusted third parties that help process transactions and keep our data secure. However, a quick scan of recent news headlines reveals that even secure, centralised systems are vulnerable to many types of attacks [1]. This leads to a need for new technologies that can guarantee trust, security to its users and carry out complex operations, where multiple stakeholders are engaged in a performant, interoperable and secure manner.

As a direct result of the above, over the last decade, many different technologies came to the surface to accommodate those needs **offering robust, secure and privacy preserving data exchanges**. However, the introduction of the concept and of the technologies of Distributed Ledger Technologies (DLTs) changed the game radically. The new concepts brought forward by these technologies allowed to not only decentralise such operations and make them more flexible, but also to deliver new standards when it comes to trusted data exchange services.

In this chapter, we provide a high-level description of why DLTs are necessary for the ASSURED framework, setting the tone for the consequent chapters that elaborate on the requirements of the overall DLT Infrastructure, the way that smart contracts operate with ASSURED and the high-level architecture that will drive the implementation phase of the ASSURED DLT Infrastructure under WP4.

2.1.1 Need for Having DLTs for Data Sharing in ASSURED

As defined in the ASSURED conceptual architecture [53], **data sharing** is one of the cornerstones of the ASSURED framework, as the overall principle for working with secure and trusted devices in a flexible SoS deployment is highly based on the ability to produce and exchange the necessary data between the different entities of the system. As such, the role of DLTs is essential, because they can provide the necessary features that is necessary to guarantee the trust that is necessary to be available in such a system, as identified in the following sub chapter, and the trustworthiness and immutability of any data sharing transaction that needs to take place over an ASSURED deployment.

When it comes purely to data sharing, DLTs are in a position to support such operations by recording transactions of various stakeholders, and are able to distinguish between transactions that need to remain visible and accessible from a sub-set of stakeholders (such as the devices belonging to an organisation that provide their attestation evidence) and of transactions that shall be made publicly available (for example exchange of information

Funded by the Horizon 2020 Framework Programme of the European Union

between different organisations for improving cybersecurity awareness). As such, DLTs do facilitate both private and public information sharing.

Moreover, the usage of DLTs allows **ASSURED to build a flexible and expandable data sharing network**. Adding new users to such networks is quite easy when adopting a DLT approach, while at the same time the addition of new users comes with the introduction of new peers into a DLT system, which strengthens the availability, and the trust guarantees of the network. Of course, access of new users/devices to the network is something that is not done automatically as the necessary credentials needs to accompany any requestor to become part of the network (in the ASSURED case these are certificates provided to the devices that specify the attributes of each device); nevertheless, once a device joins the network then the maintenance of those users is highly animated by the DLT network and the access policies that the network incorporates.

Another very important role that DLT can play relevant to data sharing under the scope of ASSURED is that of offering **smart contracts to govern the different data sharing activities that take place**. As such, it is envisaged that the different flows that will be executed during the recording, the searching and the retrieval of the information will be executed via smart contracts which will be tasked to coordinate the necessary activities that need to take place for orchestrating the complex data sharing interactions between the different entities that take part in the ASSURED DLT network and the different ledgers that will be set up. Furthermore, **smart contracts will be also used to govern the execution of attestations, clearly defining the flow of interactions between the Provers and the Verifiers and pinpoint the candidate stakeholders who should perform the verification processes**.

All this information is part of the scheduling attestation policies, compiled by the Policy Recommendation Engine, and depict the optimal set of attestation tasks that need to be executed by all devices (comprising a SoS-enabled ecosystem). The last step in the overall security process is the **deployment** of this set of targeted policies to the devices for managing the identified risks, as well as to handle the real-time supervision and monitoring of the correct execution of these policies. This is done through the ASSURED Security Context Broker (SCB) via the use of smart contracts. All attestation-related data will be recorded and kept in a traceable and accountable manner on the ledger infrastructure, thus, providing a credible security auditing and certification workflow.

However, using ledgers has also some drawbacks, which mostly have to do with performance during the execution of operations that need to be performed over the ledgers when it comes to vast amounts of data or the introduction and querying of many records. For this purpose, hybrid approaches are employed, such as having **off-chain storage facilities** where data (especially the system traces, as monitored during the execution of a remote attestation process, whose size might excess the order of KBytes) is placed and the location of those is provided as **pointers which are stored in the ledgers**. In this context, ledgers can become more performant, as small data structures are stored and queried over these facilities, while at the same time performant technologies for storing data in different types of databases or storage engines can be used.

2.1.2 Need for Having DLTs to Facilitate ASSURED Trust Guarantees

As known, DLTs high level concept resembles that of a secure functioning of a decentralised digital database. The overall approach eliminates the need for a central authority to check against manipulation. DLTs allow the storage of all information in a secure and accurate manner using cryptographic keys (though in many cases as identified above the ledgers are

Funded by the Horizon 2020 Framework Programme of the European Union

not the most performant infrastructure to hold large amounts of data). Moreover, DLTs have many other attractive features, such as **decentralisation, persistence, anonymity, and auditability**. These features make the use of DLTs a promising solution that will be adopted by the ASSURED framework to address the **security, trust and confidentiality challenges which are present in an IoT SoS deployment**. There is no central authority required that ensures the consistency of records. With each update, records are dispersed simultaneously to peer nodes, who collaboratively ensure that the updates are correct. This feature makes it possible for a ledger to be distributed among all those using it, putting the responsibility to maintain and validate it in the hands of those using it. This results in a decentralised system of data registry where transactions are transparent, reliable, and incorruptible. It's often referred to as "the trust protocol''. The use of a DLT provides us a way to trust information and data stored that may eliminate the possibility of all kinds of fraud. It also supports the download and execution of ASSURED smart contracts in a trustworthy way. As such, a DLT securely records, stores, manages, and transmits transactions in a whole host of domains. We use the term "distributed" because the record of each transaction is kept in more than one place, sometimes in thousands. Users of such a DLT must continuously agree about the current state of the ledger; if anyone attempted to alter a transaction, the network would no longer arrive at a consensus and would reject the altered record. Most notably, one of the biggest advantages of the DLT architecture, adopted in ASSURED, is its scalability, as trust is shifted from the backend infrastructure to the edge devices using trusted computing architecture through trust anchors to achieve security, privacy, and scalability.

Based on the above, it becomes obvious that the use of DLTs will enhance **trust in the ASSURED framework where devices perform attestation via the execution of smart contracts**. The attestation behaviours will be reflected into smart contract and executed by running the smart contract via the trusted platform module embedded in ASSURED devices. The attestation results will be then merged on the ASSURED DLT private ledger, and a summary of the results (metadata) will be put on the public ledger. Any ASSURED device with access rights to the private DLT can act as a verifier and, thus, will be able to run the verification algorithm from the smart contract intaking the results to check if an existing attestation is valid. The attestation results and output of verifier will be recorded on the private ledger for later auditing. A trust behaviour evaluation ranking via the use of smart contracts will be used to evaluate a DLT user's reputation/trust value to enhance the trust among users. The evaluation for the trust reputation will be based on some trust calculations and trust results will be auto executed and stored on ASSURED private ledger when needed. The regulation of the access control operations, e.g., device revocation or denying or granting some action will be based on reputation/trust scores. Based on the DLT trust scores, data access and sharing of encrypted data can be facilitated via employing Attribute Based Encryption ABE mechanisms, defined on the smart contract, and executed at the device level. Also, the Searchable Encryption SE component will be adopted in the ASSURED framework to enhance secure search on ASSURED DLT.

2.2 INTERACTION WITH OTHER COMPONENTS

As aforementioned, the main vision of ASSURED consists of two pillars: (i) the use of **attestation schemes for enhanced operational assurance** [51], and (ii) the **enforcement of dynamically adaptable policies** depicting which schemes to operate.

Funded by the Horizon 2020 Framework Programme of the European Union

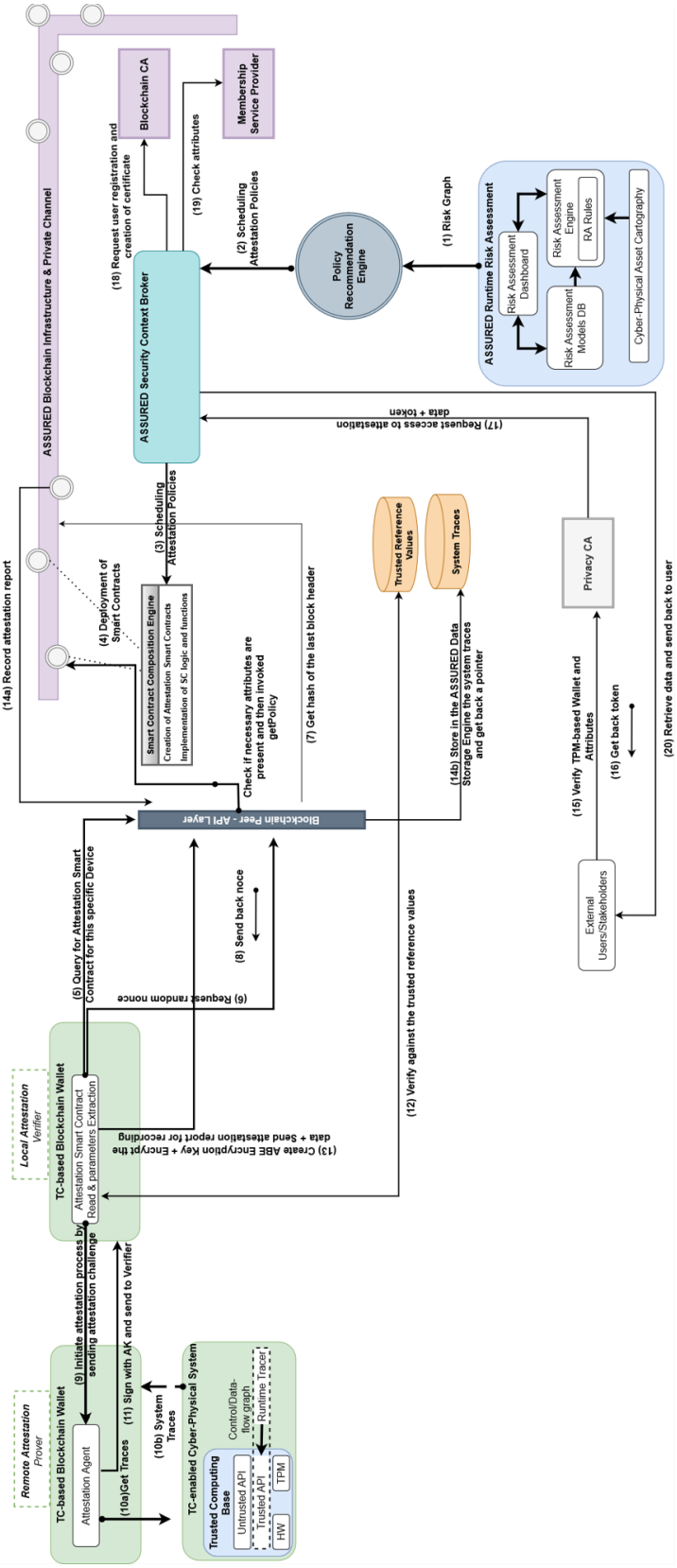


FIGURE 2: ASSURED SECURITY PROCESS AND INTERACTION WITH BLOCKCHAIN INFRASTRUCTURE

Funded by the Horizon 2020 Framework Programme of the European Union

Especially for the latter, which is enforced through the use of smart contracts and DLTs, the endmost goal is to enable the **reliable, secure and privacy-preserving extraction and sharing of knowledge (originating from verified and authenticated data sources) and threat intelligence (attestation-related) data**.

This vision is achieved through the **design and implementation of policy-compliant blockchain structures** to be enhanced with advanced **on- and off-chain data and knowledge management services** through the specification of appropriate security services including access control, smart contract composition (reflecting both the scheduling attestation policies, as produced by the Policy Recommendation Engine, but also the sharing configurations that all devices must adhere to depending on any privacy constraints), trusted consent management, membership authentication, trusted ledger and identity management (based on the use of the decentralized TPM-based Wallets) as well as privacy-preserving services.

In this context, we put forth the entire **ASSURED security process pipeline** and how all of the integrated components as well as internal devices (i.e., edge devices already part of the SoS-enabled ecosystem) and external stakeholders (i.e., administrators, certification bodies, etc.) can interact with the blockchain infrastructure and for what purpose. Essentially, we describe the **flow of actions and mode of operation of all the ASSURED blockchain-related components** and their positioning in the overall security process. *This serves as a preview of the technologies that will be used in this pipeline, and for which purpose, that will be described in detail in the subsequent chapters*. Chapter 6 provides a more elaborate view of all ASSURED blockchain-related components while Chapter 7 delves into the details of all internal functionalities and operations for the following actions.

(Step 1) Starting with the Risk Assessment Engine, this component is responsible for identifying all of the vulnerabilities and calculating the respective risks for all of the hardware assets comprising the target system. This information, constructed as a risk graph and modelled leveraging the MSPL policy language, is fed to the Policy Recommendation Engine which, in turn, compiles the optimal set of attestation policies **(Step 2)** that need to be executed in each of the assets towards enhancing the overall level of trustworthiness to the desired level. The Policy Recommendation Engine is an optimization process consisting in solving a sequence of Constraint Satisfaction Problems (CSPs): These revolve around the desired level of **security and safety per asset, the execution time per task running in each asset and the asset resources availability**. Thus, the output essentially comprises the order of execution of the specific types of attestation tasks that when employed can protect against the identified vulnerabilities.

Once the scheduling (attestation) policies have been compiled, the next step in the security process is the **deployment of this set of targeted policies to the respective edge devices** for managing the identified risks, as well as to handle the real-time supervision and monitoring of the correct execution of these policies **(Steps 3 and 4)**. This is done through the **ASSURED Security Context Broker (SCB)** via the use of **smart contracts** leveraging the designed policy-compliant blockchain infrastructure. The SCB, acting as the *trusted operator* of the produced policies, will be triggered by the Policy Recommendation Engine for converting the attestation policies into smart contract logic and further deploy the smart contract to the ledger. All attestation-related processes and data will be **recorded and kept in a traceable and accountable manner** on the *permissioned* ledger infrastructure, thus, providing a credible security auditing and certification workflow. More particularly, the SCB upon receiving the scheduling policies, it will trigger the Smart Contract Composition Engine **(Step 3)** for creating the chain code of the smart contract logic needed for the *deployment, execution and sharing*

Funded by the Horizon 2020 Framework Programme of the European Union

of the attestation tasks and results **(Step 4)**: All these operations are supported through the definition of appropriate smart contract functions as documented in D2.2 [53] and will be further elaborated in D2.5 [55] where all the functionalities of the SCB will be documented. The chain code is written in Solidity.

After the deployment of the smart contracts, edge devices, external stakeholders and the SCB can interact with them through appropriate **functions and ledger interfaces**. As will be described in Chapter 6, **Blockchain Peers** are deployed – as part of the infrastructure – offering an API Layer that provides somewhat of a "*gateway*" for bridging connections and communications between edge devices and external stakeholders with the ledger infrastructure. Therefore, what is pending is to define the necessary functions for supporting all the required operations that are needed when an entity interacts with a deployed "*attestation smart contract*".

In this context, **Edge devices** (Prover) are able to query (and read) for the attestation policies that they need to execute with a Verifier whose, in turn, will provide the trusted input to the Blockchain Peer for validating the output and merging the result of each attestation task on the ledger (thus, creating an attestation history per device and service graph chain).This process is supported by the TPM-based Wallets of both the Prover and Verifier. More specifically, the TPM-based Wallet of the Verifier (TCW_{Vrf}) also downloads the attestation policy **(Step 5)** and initiates an **attestation challenge** with the Prover. In order to provide freshness in this operation (and to avoid replay attacks), this challenge contains a *random nonce* that is generated by the Blockchain Peer, when invoked by the TCW_{Vrf}, by leveraging information from the ledger (Chapter 5.2) **(Steps 6-8)**. Upon reception of this attestation challenge **(Step 9)**, the Prover extracts the necessary system traces – from its internal Trusted Tracer **(Step 10)** – and will sign them through its TPM-based Wallet (TCW_{Prv}) **(Step 11)** prior to be sent back to the Verifier. The Verifier will, finally, verify the correctness of the traces (by comparing them to **trusted reference values** [51]) **(Step 12)** and will send the attestation report to the Blockchain Peer **(Step 13)** for one additional ratification of the correct execution of the attestation process based on the included signatures. If successful, this will then be recorded to the ledger through the Orderer component **(Step 14a)**.

We must note here that, as aforementioned, it is **only the attestation result that is recorded on the ledger** due to the monitored system traces been quite excessive in size. These are forwarded by the Blockchain Peer to the SCB which stores them to the ASSURED Data Storage and Indexing Engine **(Step 14b)** (all traces are encrypted by the devices through the newly designed Attribute-based Encryption (ABE) scheme (Chapter 6.3.2)), retrieves back a pointer which will then be amended to the created block.

In the case of a device wishing to read an attestation report, then it can query for the attestation history of a specific device. However, it needs to exhibit the **appropriate attributes and privileges** in order to be able to retrieve back the attestation related data. Attributes are verified during the Secure Enrolment phase (Chapter 6.4.2.1), by the Privacy Certification Authority (Privacy CA) **(Step 15)**, which then credits the device a token **(Step 16)** that can forward to the SCB for verification **(Step 17)**. 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 **(Step 18)** based on which it can access specific attestation result bundles. In this case, every time a device (or an external stakeholder as will be seen later on) performs a query operation then this "querying function" will interact with the SCB that will invoke the GetX509Certificate function for fetching the user's certificate from the Blockchain CA. This is then forwarded to the Membership Service Provider (MSP) (Chapter 6.2.8) that verifies that all required attributes are included in the user's certificate **(Step 19)** – in which

Funded by the Horizon 2020 Framework Programme of the European Union

case it notifies the blockchain Peer to retrieve and send back to the user/device the request attestation data report **(Step 20)**.

Additionally, external stakeholders are considered users and devices that are not part of the target ecosystem but wish to get access to a sub-set of attestation results, provided that they have the correct privileges and attributes, for certifying the overall state of the SoS-enabled ecosystem or checking the level of assurance and trustworthiness of specific devices and processes. Besides appropriate **access-control functions**, that are part of the smart contracts, this also entails been able to read the list of attributes that they need to exhibit for been able to generate the correct decryption keys for accessing the attestation results. ASSURED leverages a newly designed decentralized Attribute-based Encryption (ABE) mechanisms where the necessary encryption/decryption keys are managed locally by the TPM-based Wallet of each entity.

Finally, as aforementioned, the SCB is regarded as the trust anchor for deploying the attestation smart contracts (when the chaincode has been created by the Smart Contract Composition Engine) as well as managing all the internal processes needed for the auditable recording of the attestation results. Consider, for instance, the selection of the appropriate blockchain peers that will be acting as the “endorsers” that need to first ratify the correct execution of an attestation process (as depicted in the respective contract) prior to merging the result on the ledger.

Funded by the Horizon 2020
Framework Programme of the European Union

3 DLT AND DATA PROTECTION TECHNOLOGIES

In this chapter, we review the core platforms and building blocks we are going to investigate in the context of the ASSURED project, give a detailed analysis of the state-of-the-art and then present the enhancements that will be performed in the project towards secure on- and off-chain interactions when it comes the secure data sharing of both operational and attestation related data.

3.1 SETTING THE CONTEXT

Blockchain technology has been developed and deployed in many real-world applications, e.g., IoT and digital finance. Since the introduction of Bitcoin[1], many blockchain platforms have been proposed, e.g., Ripple[2], Cardano[3], Ethereum[4], and Cosmos[5]. A blockchain platform should be designed and developed to capture different layers, including data layer (e.g., supporting data integrity via the use of digital signature, hash, Merkel tree), network layer (e.g., developing advanced Peer-to-Peer protocols), consensus layer (e.g., the design of PoW, PoS, PBFT), contract layer, and DApps application layer. These layers mainly form the technical framework of current blockchain systems. The application layer lies on top of the framework, and it focuses on developing DApps solutions, the network layer concentrates on tackling network communication issues for blockchain systems, while the data, consensus and contract layers are used to manage blockchain data, provide incentive mechanisms for consensus and smart contract operations in blockchain. We say that the last three layers are strongly related to the ASSURED vision in offering secure and decentralized data sharing and management and leverage smart contract to enforce policy-based attestation and controlling reasonable consensus for ASSURED blockchain users.

Current blockchain platforms provide different features, e.g., in consensus, transaction efficiency and scalability, but they usually consist of the identical and core components in the data layer that run and secure the blockchain environment. These components include the block (including pointer, linked list and chain structure), transaction, hash function, Merkle tree, digital signature (e.g., elliptic curve digital signature algorithm (ECDSA), and blockchain entities (e.g., node, miners). A blockchain platform can define and design its own way to manage and run these components, for example, the platform can choose (more precisely, can be predefined and set one) consensus algorithm it uses for mining. We note that in a general context, a consensus algorithm is used to conclude an agreement on a given status within blockchain network, among all the blockchain users with validation ability, for instance, PBFT, is a type of voting consensus, if a majority voters agree on a status, then the status (e.g., client A has sufficient funds to make a transfer to client B) is accepted within the network and further merged on the ledger. Note later we will introduce that the consensus is needed in our ASSURED blockchain during the ordering services, because the orderers should conclude the final order of all the received transactions.

[1] https://bitcoin.org
[2] https://xrpl.org
[3] https://cardano.org
[4] https://ethereum.org
[5] https://cosmos.network

Funded by the Horizon 2020 Framework Programme of the European Union

In the ASSURED blockchain-based framework (which will be further introduced in chapter 6), we will require several technical components. The first need is of course a foundational blockchain platform. We will start with an initial and existing blockchain platform and further reflect and build our secure data sharing, recording and attestation-based auditing services to form our ASSURED DLT. The use of this component is to enable distributed and collaborative data sharing, storage, and management in the decentralized supply chain context. In chapter 3.1, we will review the existing state-of-the-art blockchain types and platforms. From the review, we will further identify a potential starting-point system (Hyperledger Fabric) as the ASSURED blockchain cornerstone.

The second technical component we require are smart contracts. A smart contract is regarded as a significant technique opening a brand-new era for blockchain applications. This automatic executable programme can put business logic to the ledger and protect its execution. ASSURED will leverage the smart contract to design, execute and enforce policy-based operations, e.g., data sharing and attestations. We state that this is one of the core innovations of ASSURED. We will be able to enforce the attestation policies (which are defined in the D2.2) as chaincode (i.e., smart contract deployed on ASSURED blockchain ledgers), so that we can offer timely policy enforcement, automatically and decentralised auditing the entire attestation process for all devices and then also certify the assurance process of the whole SoS ecosystem. We will review this component in chapter 3.2 to introduce its basic concept, mechanism, and advantages.

The third component we will use is the access control mechanism for the DLT network. In the following chapters, we will highlight that we prefer to use a consortium and permissioned blockchain. In this case, we will need a secure and scalable access control approach to guarantee that only valid users can access a specific channel and the corresponding ledger. To this end, credentials, and verification on them are required. In chapter 3.3, we will review a membership service provider mechanism used in Hyperledger Fabric. We say that this mechanism will help us to form the access control mechanism for our ASSURED blockchain.

In a blockchain system, while creating a block, we may require a consensus algorithm to make sure that this block can be agreed by most or majority of the validators of the network. This consensus algorithm is the fourth component we will need in the ASSURED blockchain framework. We will mainly put attestation report, storage information (e.g., pointers), integrity guarantee (e.g., hash value of reports), metadata of reports into a block. We may need a fast and secure consensus algorithm for the consortium-based blockchain to enable us to check the correctness and order of a given block (note this is done during the endorsing, ordering and validation stages). In chapter 3.4, we review the current popular consensus algorithms and further identify a potential candidate.

Recall that we will have data operations and management within our ASSURED blockchain. To secure the on-chain and off-chain data, we will further make use of two cryptographic components: one is the Attribute-Based Encryption (ABE), and the other is the Searchable encryption. We will enable devices to encrypt uploading data via the use of ABE; and while they perform the corresponding decryption, they will require the help of TPM to generate an attribute-related key. As for the searchable encryption, we will use it for the secure data search over the public ledger. We will enable the Security Context Broker to create and manage a secure and searchable index structure for metadata, so that the metadata are all encrypted but still can be searched. Another innovation to highlight in the project is that the design of lightweight ABE and searchable encryption for the secure on and off-chain interactions via help of TPM-based Blockchain Wallet.

Funded by the Horizon 2020 Framework Programme of the European Union

We note that we do not give detailed descriptions on how we use the above components in this stage. But we will present the details in chapter 6.

3.2 BLOCKCHAIN TYPES AND PLATFORMS

All existing blockchain platforms can be summarised into three main types:

- public/permissionless
- private/permissioned and
- hybrid blockchains.

A permissionless network is a public blockchain network configuration designed to allow public participation, while a permissioned network is a private blockchain network including only authorised participants. For example, Bitcoin is a classic type of permissionless blockchain, and Hyperledger Fabric is an example of permissioned blockchain. We summarise these types in Table 1. From the comparison, we initially consider using private/permissioned blockchain for the ASSURED project. This is so because we prefer to use authentication of identity to become a main mechanism to guarantee access control on ASSURED blockchain ledger. And meanwhile, we do not want to lose the decentralised and immutability features of the blockchain. The permissioned blockchain can provide high efficiency and fast transaction speed, which indicates that it can handle sufficient network tasks in a short time slot matching a consortium and a close supply chain community context.

TABLE 1: TYPES OF BLOCKCHAINS

item	Public Blockchain	Private Blockchain	Hybrid Blockchain
Access	Anyone (no authentication)	Authenticated users	Authentication & non-authentication users
Authority	Decentralized	Partial decentralized	Mixed full and partial decentralized
Transaction Speed	Slow	Fast	Adjustable
Consensus	Permissionless	Permissioned	Flexible
Efficiency	Low	High	Flexible
Data handling	Read and write access for anyone	Read and write for authenticated users	Mixed
Immutability	Full	Partial	Flexible

In ASSURED, we will consider using smart contracts to support data sharing, access control, attestation, and other operations (see ASSURED Deliverable D2.2 [2]). We then should turn our attention to those permissioned blockchain platforms that support the deployment and execution of smart contract. From this perspective, we should mainly focus on the well-studied and relatively mature platforms, namely Hyperledger Fabric[6], Quorum[7], R3 Corda[8] and MultiChain[9].

Hyperledger Fabric (HLF) [3], which is an open-source and one of the most popular

[6] https://www.hyperledger.org/use/fabric
[7] https://consensys.net/quorum/
[8] https://www.r3.com/corda-platform/
[9] https://www.multichain.com/

Funded by the Horizon 2020 Framework Programme of the European Union

permissioned blockchains, was invented by IBM and further developed by the Hyperledger Foundation, being applicable to enterprise level blockchain development. HLF can provide the use of smart contracts (called chaincode) to implement logic, with general-purpose scripting languages, e.g., Go, Java and Node.js, instead of those limited domain-specific languages. This brings convenience to smart contract development, and further enables blockchain users to execute more sophisticated business logics, e.g., attestations. Beyond this, HLF can also support so-called pluggable consensus algorithms (e.g., PBFT, Raft[10], Kafka[11]), so that one may tailor its needs to select which consensus protocol to use. Besides, it can offer privacy for communication via the design of channels, private data control via the access control list, and membership mechanism to restrict channel access. HLF consists of network nodes and their identities are managed by the membership service provider. These nodes can be any of the following: clients - proposing and broadcasting transactions, peers - maintaining the ledger and state (participating into execution and validation processes in compliance with endorsement policy); or ordering service nodes - which only define the order of transactions on the ledger.

Quorum [4], developed by J.P. Morgan and deployed in the financial sector, is an enterprise-focused and open-source variant built on top of Ethereum - a fork of Go-Ethereum, which makes it incorporate the Ethereum network for update seamlessly. Quorum is fast in transaction settlement since it is using vote-based and different algorithms to take hundreds of transactions per second. It can handle applications requiring high throughput processing and speed of transactions. And it provides private and public on-chain transactions/contracts to maintain the confidentiality of records. It also supports flexible consensus protocols like Raft-based and Istanbul BFT.

In 2015, R3 built an open source blockchain called Corda [5]. Corda enables different business entities to transact via smart contracts (written in JVM languages) without costly frictions from transactions. And it uses the authorisation to enable users to access data via the design of “Doorman” - a node taking the role of identity validation and certificate distribution. It enhances privacy and offers fine-grained access control to digital records. Its privacy protection concept is maintained by showing only the minimum amount of information and 'tearing-off' the information that should be kept confidential from the transaction.

Designed by Coin Sciences, MultiChain [6] is seen as a fork of Bitcoin and allows one to set up private ledgers with efficiency, offers a command-line interface for network interaction and extends the core functionality of Bitcoin API. Providing a flexible level supports to programming, it enables different clients such as C#, Go, Java, Node.js, PHP, Python and Ruby to interact with the network through JSON-RPC API. MultiChain uses its special round robin validation as a consensus algorithm, in which blockchain nodes are pseudo-randomly selected to create blocks, but a node must wait several block-creation cycles before being chosen again to add another new block.

These permissioned blockchain platforms can use public key infrastructure and support private transactions. HLF enables one to configure privacy via channels and private data access. In this case, a completed transaction’s data can be protected and only accessed by some specified users, in which everyone in the channel can see the transaction but not its data. We state that this feature is extremely useful in our ASSURED context. With this, blockchain users can notice that an attestation of a device has taken place, but they cannot read the attestation result unless they are granted the corresponding privileges and attributes. Like the design of

[10] https://raft.github.io/
[11] https://kafka.apache.org/

Funded by the Horizon 2020 Framework Programme of the European Union

HLF, Quorum and MultiChain use private transactions to hide the transaction data so that limited users can access them, where these transactions are still broadcasted to the whole network. But in Corda, its transactions can also be private, and the credentials are kept confidential to hide transaction identities, and further, since Corda cannot support global state, transactions can be set to only seen by some users rather than the whole network. This may bring some risks for the transactions because they cannot be verified by network nodes.

As for scalability, HLF can outperform others because it may provide various types of consensus mechanisms and interfaces. It can provide us open-source platform for R&D, pluggable consensus algorithm options, smart contract implementation, fast performance, and interface to support trusted hardware, e.g., TPM. In this way, one may have better options while dealing with different application contexts and adapting better to scenario switching. Quorum and Corda may be able to support more than one consensus, but MultiChain is restricted to one type of consensus: the round robin validation. Furthermore, HLF achieves a higher throughput and smaller latency than other platforms. Thus, HLF should be the best option so far due to its privacy, scalability, and performance features, and it is also open sourced for re-developing. And it may be the best to be compatible with the needs of our ASSURED DLT platform and secure data cryptographic tools. We summarize the comparison in the following table.

TABLE 2: MAIN BLOCKCHAIN PLATFORMS COMPARISON

Blockchain Technology	Support	Open Source	Smart Contract	Consensus	Privacy	Scalability	Performance
HLF	IBM and Hyperledger Foundation	yes	yes	Solo, Pluggable	Channels, private data access control	Plug-in consensus	best latency and throughput
Corda	R3	yes	yes	vote-based or Raft	private transaction	two options	good latency but poor throughput
Quorum	JP Morgan	yes	yes	PoA, Raft, Istanbul BFT	private and public transactions	three options	good throughput but poor latency
MultiChain	Coin Sciences	yes	version 2 yes	Round Robin validation	private and public transactions	hard-coded one option	good throughput and good latency

3.3 SMART CONTRACTS

Smart contracts were first introduced by Szabo [7] and they were known as protocols that could be auto run and enforced without any help of trusted third parties. Since the introduction of blockchain, smart contracts have been defined as event-driven computer programs defined, executed and enforced by the participants when a certain event happens, based on specified parameters [8] in a blockchain network. Ethereum introduces a brand-new virtual machine structure to support the Turing-complete programming languages that enrich smart contract functionalities. The smart contracts then can be written in high-level programming languages, e.g., Solidity, and then compiled into the EVM bytecode. Inspired by the design of Ethereum, many popular blockchain platforms, e.g., Quorum and Hyperledger Fabric, have provided smart contract implementation.

Funded by the Horizon 2020 Framework Programme of the European Union

Smart contracts basically work by following the "*if/when…then…*" pattern. The logic implemented by a smart contract is domain-specific and its source could be from some law documents, bi-agreement between users, and other business process requirements [9]. We usually require public interfaces to handle the relevant events. The interfaces can be invoked by the transactions with proper payload data, and all valid transactions are recorded on the blockchain. A smart contract will be first designed and deployed on the blockchain, and then a unique address will be assigned to identify the smart contract. The specified blockchain users can invoke the smart contract by sending a transaction with the address. This transaction will be later handled by a miner who executes the code of the smart contract and then performs actions on the specific tasks written on the contract. And later, the produced results will be updated to the blockchain ledger. Only parties who have been granted permission can see the results.

Using smart contracts in real-world blockchain-based applications yields many benefits. The contracts can provide a speedy and accurate execution of pre-specified tasks. Once a predefined condition is satisfied, the contract should be executed immediately. There is no tedious paperwork and extra time spent on manually checking or third-party involvement. The smart contracts can provide trust and transparency between contract parties. They are only executed, and the results can be read by particular parties, and the records are traceable since they are merged on the ledger. And, the contracts reduce the time cost and fees incurred by a trusted third party, and there is no need to worry about the third party may bring bias, delay or extra fees for handling the execution of the contract. Due to these advantages, smart contracts can be used in digital payments, digital identity management, estate transfer, insurance, and supply chain applications.

In the ASSURED project, we will make good use of the smart contract to support attestation, policy checking and data sharing operations. However, we do not consider countermeasures on the attacks on smart contracts, e.g., re-entrancy, front-running, which means that we assume the smart contracts we deploy and use in the project are secure and not corrupted in the consortium based blockchain network. We say that this assumption makes sense because all the consortium partners are authenticated and known by all partners. But we will still support certification and authentication by the Privacy CA who will check the validity of the TPM wallet of each device, and the wallet can be used to manage the interaction with smart contracts. In the ASSURED blockchain context, a misbehaved partner will be removed and cancelled its credential to the blockchain network easily by the Blockchain CA.

We note that in general setting a smart contract will be converted to a form that can be understood and read by blockchain node, and further the node will deploy it in the blockchain network. Later, a blockchain client can call the smart contract for execution and the resulting output will be merged on the blockchain ledger. We will put these details in Chapter 5.

3.4 ACCESS CONTROL MECHANISMS

As mentioned previously, in the ASSURED project, we may prefer to make use of the HLF as our system blockchain foundation. And in the HLF, there exists an efficient and secure access control mechanism which is called membership service provider (MSP). In a blockchain network, there are usually several different entities, e.g., peers, orderers, clients, and each entity should have its own and distinct identity. These identities should be issued by trusted blockchain certificate authority (CA) along with the CA's signature on it. We note that in the context of ASSURED, we will use two different kinds of CA: one is called the Blockchain CA and the other is the Privacy CA. The functionalities of them are quite distinct. The former is for

Funded by the Horizon 2020 Framework Programme of the European Union

the credential generation and verification for the blockchain users, while the latter is used for the TPM attribute verification and a registration token generation.

Any new devices are first verified by the Privacy CA. After certifying their TPM wallets and their attributes, the Privacy CA will provide them with a token that can then be sent to the Blockchain CA for creating the devices' credentials. As for these credentials, the private/public key pair is then sent to and stored on each device's TPM wallet. After the above enrolment, the MSP will further manage all the credentials within the entire blockchain network. All entities should have to be verified their identities before performing any blockchain based operations, for example, a client should sign a transaction before sending it to the peer, and the endorsers and orderers will need make the corresponding signatures on the endorsement and the new block later. Each entity within the blockchain network should be provided by MSP, and the corresponding credential, such as private key, certificate, should be protected locally by the entity. In the ASSURED blockchain framework, we use TPM as a trusted and secure anchor for the credential.

And further, the TPM enables the entity to fulfil secure operations, including transport layer security for the communication. Each user should receive an identity and a certificate. The certificate has two parts, one is a private key for digitally signing and being stored securely in TPM. The other is a public x.509 signcert certificate. That includes a public key and other attributes provided by CA. We note that the certificate is allowed to have various attributes, including organisation details, certain roles, authorisation, and other information. In the ASSURED context, we will set the attributes to be those are mainly for who can read which attestation policies and who can access to the attestation results.

There are two types of MSP in the HLF network: the local MSP and the channel MSP. The *local MSP* is for clients and for peers and orderers**.** It can be used to define the permissions for a blockchain node. We note that a traditional blockchain node is used as an operational “manager” for getting input and output of transactions, validating transactions, and merging them on blockchain ledger. In the ASSURED context, we call it peer which can access input/output of attestation-based smart contract functions, and further record them on ledger. The local MSPs of clients, allow the user to authenticate itself in its transactions, or as the owner of a specific role into the system, e.g., an organisation admin. Peers and orderers must have public and private keys and the corresponding signed certificates from CA. In these entities, they are allowed to equip with several artefacts, including a list of administrators’ certificates, the CA public cert for verification, and a list of revoked certificates. With these digital documents, they can perform valid authentication on blockchain entities, for example, if a transaction is signed by a valid and claimed client. Note that the peers and orderers will later receive the channel MSP information, so that they can perform the authentication in a channel wide manner. For example, a peer could become a peer within a channel for two organisations, and it will have to verify all the transaction ownerships from the organisations; to do so, it must know the channel MSP info.

A HLF network channel usually maintains many organisational MSP information, such as which peers and orderers are joining or leaving. All this information will be dynamically configured in the *channel MSP*, and the information can be read by all peers and orderers, so that every entity can share the same view of the channel MSP. For example, a new peer joining the channel can find out who else is also in the channel. The channel MSP also configures who can connect to the channel and perform what types of operations, e.g., transactions. All the configuration is stored on ledger via transactions. The channel MSP should include public certificates of administrators, CA public certificates for the current MSP, and list of revoked

Funded by the Horizon 2020
Framework Programme of the European Union

certificates. The information is all public and does not include any private keys for identity or private information.

A new user registration. When a new user would like to register itself to the HLF network, it must communicate with the network administrator. The administrator uses the registration function supported by the Fabric-CA-Client to generate an identity (e.g., enrolment ID) and some related attributes. We note that in the ASSURED context, we will mainly use the security context broker as the network admin. This is so because it will monitor and interact with internal and external entities enrolment, data sharing and data access operations. We further note that the ASSURED will make use of an extra Privacy CA for the authentication of new user in the blockchain network, which has been mentioned previously. The registration will be first gone through a Privacy CA and then the (Blockchain) Fabric CA, in particular the device's TPM related attributes should be verified by the Privacy CA, so that a token is generated from the Privacy CA for the Fabric CA to create blockchain credential for the device. The purpose of using these two CAs in ASSURED is that we can provide a security check on trusted hardware TPM, and then further use this trusted anchor to proceed the blockchain credential generation. If we do not use the Privacy CA, then no one knows that if a given TPM is secure, which means that a trusted anchor assumption cannot hold.

The Fabric-CA will generate secret information as well. The administrator will pack this information and send it to the user via a secure channel. With this information, the user can further perform a valid enrolment with the Fabric-CA providing the ID and the secret over a secure channel. The user will be given a pair of keys: a private key, and a public key signed by the Fabric-CA (with x.509 certificate). And then it will store the credential securely in its local MSP - in the ASSURED project that will be stored within the TPM. We note that the above user could be a new device, peer or orderer in the network.

Operations verification. In the HLF, the credentials of any user (e.g., a client, an orderer) can be used to provide operation verification to check if a given operation is done by the claimed identity. For instance, a client (e.g., a device) may sign a transaction proposal using its private key (stored in local MSP) - in ASSURED context it will be also signed with the TPM AK - before sending it to the peer. After receiving this transaction, the peer first validates the signature by using the corresponding CA cert (included in the information of the client's public key) stored in its list. If the signature is valid, the peer may play the role of endorser to execute the transaction, generate the results and further sign the results with its private key. The results and the signature will be returned to the peer who will verify the signature using the public known channel MSP (where the credential of peer is stored). Similarly, if the signature is valid, the client will further sign the endorsed package and send it to the orderer. During ordering service, the orderer will validate the client signature, further pack the transactions received from other clients into a block, and at last sign the block. This block will be sent back to the peer who will have a final validation of the signature and the block before merging it into the local ledger. From the above flow, we see that each entity in the HLF network should have its own local MSP to pack private and public key (credential); the operational nodes, like peers, orderers, they will maintain a list for the clients' certs for validation of their identities and attributes; and the channel MSP should also be provided so that the entities can know who are playing which types of roles within the channel, e.g., who is the peer and who will be the endorser for a transaction.

Funded by the Horizon 2020
Framework Programme of the European Union

3.5 CONSENSUS ALGORITHMS

In any blockchain platform, one should tackle a core problem that is called consensus - *how to reach an agreement among a quorum of blockchain nodes*. Since there is no centralised node in the blockchain network, it is challenging to fulfil this purpose. While distributed and decentralized networks may incur corruption and errors on network participants, we need a reliable and secure consensus protocol. In the literature of distributed systems, there is a classic problem - Byzantine Generals Problem (BGP). The problem requires a group of generals to formulate a final plan for attacking a city while they are on different sides of the city. They must agree on a coordinated attack or retreat. But in the group, there may exist some malicious generals, and then the loyal and honest ones must make sure the formulated plan can reach expected results with some rules. The above case may similarly happen in the distributed and trustless blockchain network, in which the generals are the network nodes. In this case, this requires that a blockchain-based consensus algorithm should be able to deal with Byzantine Fault-Tolerant (BFT). There are mainly two types of BFT based consensus algorithms. One is known as Proof of something, and the other is based on byzantine agreement. The formal mainly relies on a special “proof” to elect a leader to validate and generate a block; while the latter needs a quorum-based agreement (usually requiring honest majority) on the validation of transactions, for example, a group of network nodes validate transactions and all of them agrees on the statues of the transactions, and further all the transactions can be merged into a new block. Below We review the most important and popular consensus protocols.

Proof of Work (PoW) is the consensus algorithm used in Bitcoin [10]. It requires one who would like to become a valid miner to find a hash puzzle, and the first one completing the task will become the miner to perform a blockchain block mining. The algorithm requires a prover and a verifier: the former finds the puzzle and presents it to the verifier, while the latter validates the answer in a simply and fast way. In practice, a miner as the prover can mine a block that can be efficiently validated by any verifier. The PoW can be bound to a parameter to set its puzzle finding difficulty. Equally allowing everyone within the network to have a chance to become miners, the resource-intensive PoW is usually used in public blockchain platforms to avoid double spending and against 51% attacks (w.r.t. controlling 51% of computational power), sybil attacks, e.g., BitCoin, but it does need the participants to consume expensive computational resources to locate a valid hash puzzle, in which the block mining will take long processing time (as compared to other consensus algorithms). [example: Bitcoin, Litecoin[12], Dogecoin[13], Dash[14], Zcash[15], Monero[16]]

Proof of Stake (PoS), unlike PoW, is an energy and resource friendly consensus algorithm, and it requires participants - who would like to participate in the block creation - to prove ownership of the amount of currency (which is called stake, as a kind of guarantee or a collateral) in advance. And the selection of block creator will be done from a group of stakeholders. if this creator successfully completes the mining, it will be rewarded by a certain amount of transaction fee or currencies. But the selection, based on PoS, is formed as a formulation related to the account balance, which may make the rich get richer - being dominant - within the network. Many current PoS solutions consider additional factor to “fair”

[12] https://litecoin.com/
[13] https://dogecoin.com/
[14] https://github.com/dashpay
[15] https://z.cash/
[16] https://www.getmonero.org/

Funded by the Horizon 2020 Framework Programme of the European Union

the selection, for example, Blackcoin [11] injects randomisation on a group of stake owners, Peercoin (peercoin.net) takes the coin's age into account. 51% attack is believed to be economically infeasible under PoS but nothing-at-stake may be the major shortcoming. [example: Peercoin[17], Gridcoin[18], Nxt[19], Ouroboros[20]]

Delegated PoS (DPoS) is a variant of PoS based on voting-based consensus algorithm. It requires an electoral progress to a group of directors, to limit the board members for election. Besides, delegates can be selected to exercise the members' rights. Under DPoS, transaction validation, block creation, and other network operations could be completed by a group of elected delegates with corresponding reward or vote-off-list and stake-off punishment. The DPoS avoids nothing-at-stake problem, long-range attack and weak subjectivity which are intrinsic inside PoS. But it is very easy for DPoS to tend toward centralisation, allowing those members with large stakes to become validators. And DPoS may suffer from other issues, for example, validators may buy votes. [example: BitShares[21], Steemit[22], EOS[23], Lisk[24], Ark[25]]

Practical Byzantine Fault Tolerance (PBFT) [12] is built to find a secure consensus among 3f+1 participants, where f participants may act byzantine, i.e., being faulty or malicious. This algorithm was first introduced to tackle the Byzantine Generals Problem and later refined for distributed network context. The elected leader (a.k.a. primary node) orders transactions and proposes the block and the other validation nodes engage in multiple all-to-all message phases, to ensure the leader acted consistently and to agree on the resulting block. They will compute the corresponding hash value for a block. As long as ⅔ hash values are the same, the block can be committed. PBFT can enhance the throughput and it has variants, e.g., delegated BFT - for country's governance system, federated Byzantine agreement - for handling transactions among tokens issued by different entities, and redundant/simplified BFT [example: Tendermint[26], Hashgraph[27], Hyperledger Fabric, Ripple[28], Stellar[29]]

Proof of Authority (PoA) [13] can be seen under the PBFT umbrella. It enables trusted nodes to be selected to become validators for validating transactions and blocks, and it only requires a limited number of validators. These validators are not required to have stakes but just reputation within the network, which avoids the case that "the rich" will control the network sooner or later. PoA provides high throughput, scalability and zero fee for processing, but it may have a potential centralisation problem - the network will be controlled by those reputed validators. [example: Ethereum Kovan[30]]

Proof of Elapsed Time (PoET) [14] was introduced by Intel in its Hyperledger's Sawtooth project. Instead of using computing resources or stakes for competition, PoET designs a random waiting time count-down mechanisms for participants - each node should generate a

[17] https://www.peercoin.net/
[18] https://gridcoin.us/
[19] https://www.jelurida.com/nxt
[20] https://cardano.org/ouroboros/
[21] https://bitshares.org/
[22] https://steemit.com/
[23] https://eos.io/
[24] https://lisk.com/
[25] https://ark.io/
[26] https://tendermint.com/
[27] https://github.com/hashgraph/
[28] https://ripple.com/
[29] https://www.stellar.org/
[30] https://kovan-testnet.github.io/website/

Funded by the Horizon 2020 Framework Programme of the European Union

wait time, and the node with the shortest waiting period becomes the current block leader/creator, in which this party should present a proof of its shortest time and it did wait for the end of this time slot. This mechanism is more decentralised and provides friendly and low-cost resource consumption for participants, and it is safeguarded by trusted hardware Intel Software Guard Extensions (SGX). But it may not be applicable to public networks because not every node can be installed to the SGX beforehand. [example: Hyperledger Sawtooth]

Proof of capacity (PoC) [15] shares a similar philosophy of PoW using available hard drive space to identify the mining rights and transactions validation. Due to its efficiency, PoC is regarded as an alternative for PoW and PoS. It stores a list of possible solutions to cryptocurrency hashing problems on mining device's hard drive even before the mining; in this case, a hard drive with larger space can have more chances to find a correct solution to become a miner. But this mechanism is easily restricted to hard drive capacity and possibly affected by malwares. [example: Storj[31], Burst[32], Chia[33], Space[34]]

Inspired by Paxos, **Raft** [16] was introduced to maintain performance and correctness for distributed consensus. It is built on a leader-driven model, in which a leader is elected to be responsible for cluster message management - handling replication across all nodes within the cluster. Raft is an easily understandable consensus algorithm with lightweight and safe leader election, and it is comparatively easy to implement in networks requiring a minimum quorum size N/2 +1, in which N is the number of nodes in the network. It is known so far that Raft is the only crash fault tolerant (instead of byzantine fault tolerant) consensus algorithm in the literature.

HLF designs its own special consensus mechanism for ordering and validation services. It uses orderers to create a new block via organising the appropriate order of given transactions. All network parties in HLF should be registered via the membership service provider. And after a transaction is broadcast within a valid channel, each endorser for this transaction should validate and execute it and further returns the result. This information are encoded into transactions and sent to orderers who will create a block and return it back to endorsers. If the block passes validation, then it will be attached to the ledger and the state of the ledger will be updated. In this mechanism, the ordering and validation services can be supported by the SOLO, Kafka, Raft and simplified BFT consensus algorithms.

Based on the above review, we could summarise the pros and cons of these popular consensus algorithms in Table 3. From the comparison, we may identify a few possible candidates for our ASSURED blockchain platform. We will focus on permission-oriented consensus algorithms with low energy consumption. This leaves only PoA, PoET, PBFT, RAFT and Fabric. Since we may not consider using trusted hardware as a root for a consensus algorithm, we will ignore PoET.

The rest of the algorithms are quite related to BFT and meanwhile, it seems that HLF is a central balance point providing feasibility and interfaces to support BFT based consensus algorithms. From this perspective, we will use Raft which is the crash fault tolerant mechanism as a starting point.

[31] https://www.storj.io/
[32] https://www.burst-coin.org/
[33] https://www.chia.net/
[34] https://spacecoin.network/

Funded by the Horizon 2020 Framework Programme of the European Union

TABLE 3: THE COMPARISON OF CONSENSUS ALGORITHMS

Consensus	Blockchain type	Mechanism	Fault Tolerance	Energy consumption	Throughput	Example
PoW	permissionless	Lottery, randomised	2f+1[35]	High	Low	Bitcoin, Peercoin, Dash, Zcash
PoS	permissionless	Lottery	3f+1	Low	High	Nxt, Gridcoin, Ouroboros, Algorand[36]
DPoS	permissionless	Voting	3f+1	Low	High	EOS
PoB	permissionless	Probabilistic lottery	2f+1	medium	medium	SlimCoin[37]
PoC	permissionless	Voting	N/A	High	Medium	SpaceMint
PoA	permissioned	Voting	3f+1	Low	High	Ethereum Kovan; Polkadot[38]; VeChain[39]; Aura[40]; Clique[41]
PoET	permissioned	Lottery	N/A	Low	High	Hyperledger Sawtooth[42]
PBFT	permissioned	Voting	3f+1	Low	High	Fabric, Diem[43], Tendermint[44]
Raft	permissioned	Voting	n/2-1	Low	High	Fabric

3.6 ASSURED SECURITY AND CRYPTO PRIMITIVES FOR SECURE DATA MANAGEMENT

3.6.1 Dynamic Searchable Symmetric Encryption

Since the first CKA2-secure DSSE scheme in 2010 for which the search efficiency is logarithmic with the number of unique keywords [17], some formalised work by Stefanov et al. [18] and Bost et al. [19], [20] have shifted efforts to develop DSSE schemes with forward and backward security.

In 2016, Garg et al. constructed a forward-secure DSSE scheme based on their **TWORAM** [21]. In 2017, Kim et al. utilised the dual dictionary to construct a forward-secure DSSE scheme that supports real deletion [22]. In 2018, Song et al. proposed a counter-based forward-secure

[35] where f < n/3, where f the number of Byzantine nodes and n denotes the number of total nodes participating in the network. The fault tolerance in PoW is tied to computational power as well.
[36] https://www.algorand.com/
[37] https://slimcoin.info/
[38] https://polkadot.network/
[39] https://www.vechain.org/
[40] https://auraluxuryblockchain.com/
[41] https://github.com/ethereum/go-ethereum/tree/master/consensus/clique
[42] https://www.hyperledger.org/use/sawtooth
[43] https://www.diem.com/
[44] https://tendermint.com/

Funded by the Horizon 2020 Framework Programme of the European Union

DSSE scheme **FAST/FASTIO** with real deletion support [23]. Their proposed scheme achieves I/O efficiency by caching historical search results.

In 2016, Hoang et al. presented forward-and-backward-secure **DOD-DSSE** [24]. The core idea of **DOD-DSSE** is to let the client fetch all related data from the server and to perform Search or Update operations locally. In 2017, Bost et al. [20] constructed four forward-and-backward-secure DSSE schemes: **Fides**, **Diana_{Del}**, **Moneta** and **Janus**. Subsequently, Chamani et al. [25] proposed three improved constructions, including Type-I scheme **Orion**, Type-II scheme **Mitra** and Type-III scheme **Horus**. At the same time, Sun et al. [26] proposed a practical Type-III scheme **Janus++** by making use of their symmetric puncturable encryption. In 2019, Li et al. constructed a forward-and-backward-secure DSSE **Khons** with the hidden pointer ciphertext structure and partition search technique. In 2020, He et al. [27] presented a forward-and-backward-secure DSSE scheme **CLOSE-F/CLOSE-FB** with constant client storage. He et al.'s approach is to combine the counter and chain structure and use the global counter to find all chain structures of ciphertexts.

In the same year, Demertzis et al. [28] proposed three forward-and-backward-secure DSSE schemes, **QOS**, **SD_a**nd **SD_b** with constant client storage. The first two schemes mentioned in [28] achieve interactive real deletion, and the third scheme uses an oblivious map and a tree-based encrypted index as building blocks. Amjad et al. [29] proposed several schemes with all types of backward privacy by leveraging the power of Intel SGX. They are all non-interactive but depend on the security and reliability of trusted execution environments. Very recently in 2021, Sun et al. [30] proposed a forward-and-backward-secure DSSE scheme **Aura**, which achieves non-interactive real deletion in the cost of extra client-side storage resources to stash Delete queries.

In the ASSURED project, we will prefer to make the research works [28] and [29] as our starting point. This is because both provide high search efficiency and low client storage cost. We will see how to leverage the features of them and to possibly design and develop a DSSE scheme to maintain strong search privacy and data security but also highly scalability and practicality while merging the DSSE with our ASSURED blockchain framework, in the public ledger's secure metadata search. Intuitively, we will enable the security context broker to construct keyword-based index structure for attestation metadata and the corresponding attestation reports' storage pointers, and this structure can be further dynamically expanded as the increase of reports. The structure and the pointers will be further encrypted and stored on an ASSURED public ledger. For each external party who is trying to search over the encryption, the security context broker will deliver it a search token corresponding to some given keywords, and with the token, the party can locate the required encrypted pointers. And we state that the design details will be provided in the Deliverable D4.3.

3.6.2 Attribute-based Encryption

Sahai and Waters [31] first presented the ABE scheme in 2005 to provide flexible access control over encrypted data, which enables one-to-many encryption. There are mainly two kinds of ABE schemes: key-policy ABE (KP-ABE) scheme and ciphertext-policy ABE (CP-ABE) scheme. Goyal et al. [32] first introduced the KP-ABE scheme that supports any monotone access policy. Bethencourt et al. [33] first provided a CP-ABE scheme, which was proven secure in the generic group model. The time-consuming pairing computation in most ABE schemes is linear with a complex of access structures. Recently, some outsourced ABE schemes [34] were proposed, which improves computation efficiency. After that, some ciphertext policy hiding CP-ABE schemes [35] were introduced.

Funded by the Horizon 2020 Framework Programme of the European Union

Liu et al. [36], [37] presented black-box and white-box CP-ABE with user accountability respectively. Both schemes were constructed by using bilinear groups of large composite order, which made the schemes inefficient. To improve efficiency, Ning et al. [38], [39] presented two white-box CP-ABE schemes with traceability based on bilinear groups with prime order. To address the problem of authorisation centre key abuse, Ning et al. [40] constructed white-box traceable CP-ABE schemes with accountable authority that supported any monotone access structures. To catch those people leaking their access credentials, Ning et al. [41] presented a traceable CP-ABE scheme.

Okamoto and Takashima have developed fully secure schemes under the DLIN assumption on symmetric maps which support many attributes [42], but theirs is not a large-universe construction in the standard sense. Attrapadung has recently proposed some large universe constructions on asymmetric maps under q-type assumptions. In a recent work, Chen et al. [43] gave encryption schemes not just for ABE but a variety of other predicates like inner product, building on the predicate encoding and dual system group abstractions. Based on that work, Agrawal et al. [44] proposes the first fully secure ciphertext-policy and key-policy ABE schemes FAME, based on a standard assumption on Type III pairing groups, which do not put any restriction on policy type or attributes.

In ASSURED, we will choose to start with the KP-ABE direction. As compared to the CP-ABE, KP-ABE is able to provide less computational and communication cost for the ciphertext, and it enables users to leverage keys associated with attribute description to match those ciphertext with attributes for decryption. We will not use any of the above as cornerstone for our ABE component. Instead, inspired by research works [45] and [46], we will invent a novel ABE scheme for devices to complete efficient and fast encryption and decryption. And the KP-ABE used here will be fully done in a completely decentralized manner by using the TPM-based wallet for the creation of encryption and decryption keys as described later in the chapter 6.3.2, which is one of the main innovations of ASSURED. For more details, please see chapter 6.3.2.

3.7 SUMMARY OF ANALYSED TECHNOLOGIES

We summarize the above technologies review into the following table. In the Table 4, one may see the motivation of our choices and the initial idea of how to use the techniques into our ASSURED blockchain framework. We note that all these components as well as TPM-based trusted wallet will be introduced in detail in the following chapters.

TABLE 4: SUMMARY OF TECHNIQUES

Component	Needs for ASSURED	General use in ASSURED
HLF consortium based blockchain	We will fulfil data sharing, management, and attestation operations for the SoS ecosystem in the context of distributed and decentralized context. ASSURED requires that a blockchain system should be able to support the use of trusted hardware, flexible and fast consensus algorithm, smart contract interface, and secure access control mechanism within the blockchain network. The	In the ASSURED, we will merge the followings with the HLF: (1) enhance the access control to become an ABAC - attribute-based access control mechanism to maintain the ASSURED blockchain users valid access rights on channels and ledgers; (2) combine with smart contract interface to execute and enforce attestation based smart contract; (3) implement fast and secure consensus algorithm mainly for ordering and validation stage of attestation reports; (4) provide interface to support TPM-based wallet for cryptographic key management, authentication and attestation; (5) support secure data protection, access and search via ABE and searchable encryption.

Funded by the Horizon 2020 Framework Programme of the European Union

	features of open-source, flexible and pluggable interfaces provided by the HLF may provide us with convenient and swift R&D.	
smart contract	In ASSURED, we will need smart contracts to provide automatically and decentralized attestation operations and data accessing.	Smart contract functionalities are defined in D2.2, and in WP4 use of a policy recommendation engine and smart contract composition engine is done to support the security context broker to allow it to deploy policy-based attestation smart contract on private ledger. The smart contract will be merged into a chaincode that enables attestation proof and verification, policy generation/update, and other data access operations.
Raft consensus	We will need a fast and secure consensus algorithm to safeguard the final results from the validation and ordering stages.	Specifically, this consensus algorithm will be used in the ordering services so that a new block created by orderers could be finalized and validated. At the same time, we will also use a Jury-based attestation verification, and Raft will be also used in this type of verification to conclude a final validation on an attestation verification report.
ABAC	ASSURED requires secure access on private and public channels/ledgers for all the network entities including devices, and the blockchain entities.	ABAC will include the collaborations among Privacy CA, Blockchain CA, the security context broker and the blockchain peers/orderers/clients. The two CAs are used to generate token and credentials for devices. The security context broker is regarded as a general role of admin to check devices' policies and monitor the CAs operations. Later, accessing blockchain channels and ledger must be verified via the devices' credentials. Any operations within the ASSURED blockchain network will be signed by the entities' credentials, so there must be verifiers to check the validity of the signatures, for example, a device should be authenticated using its blockchain credential via a peer before granting access to the ledger, and the device should check if the execution results of smart contract sent back by the peer is signed by the peer.
ABE	ASSURED requires data confidentiality and meanwhile the protected data can be fine-grained accessed via the use of attributes, in the private ledger.	In ASSURED, we will enable devices to perform ABE to encrypt the attestation report log using attributes, and while decrypting the encryption, we enable TPM to help device to generate the corresponding attributes key to recover the full attestation contents.
Searchable Encryption	ASSURED requires that any external parties are able to perform fast and secure search over the public ledger to locate any related attestation files.	In ASSURED, we will implement this need via the user of searchable encryption. In general, the security context broker will pack the searchable and encrypted structure for the metadata of attestation reports on the public ledger, while an external party would like to search over those encrypted data, it will require a search token from the broker. A matching encrypted pointer corresponding to the token will be returned after search. We will consider using this as a dynamic mode to enable the increase of metadata from various attestation reports.

Funded by the Horizon 2020 Framework Programme of the European Union

4 ASSURED BLOCKCHAIN RELEVANT REQUIREMENTS

Chapter 3 provides the list of requirements which are relevant to the ASSURED blockchain infrastructure that aims to support the overall concept described above. In this context, as the ASSURED DLT infrastructure aims to facilitate the secure and trusted sharing of data (either operational or threat intelligence data) between different entities, the overall processes followed to elicit these requirements was based on analysing and elaborating on:

- the already defined high level data sharing requirements of ASSURED, from the perspective of a blockchain system to be deployed. These are listed as the DLT Sharing Requirements and are presented in chapter 3.1.
- the data security, confidentiality and trust guarantee requirements that are relevant to the different operations to be provided by the different components that will take part in the overall ASSURED framework. These are the DLT Data Security, Confidentiality and Trust Requirements and are presented in chapter 3.2.

The complete set of requirements defined in this chapter will act as the material to define the core elements of the DLT architecture which is presented under chapter 6 and will also define the different operations and data flows that are described under chapter 7 of the current document.

4.1 DLT SHARING REQUIREMENTS

As identified in chapter 3 of the ASSURED deliverable D1.4 “Report on Security, Privacy and Accountability Models” [47] where the different data sharing profiles have been defined, in ASSURED we envisage 4 different types of data flows, namely those of main business data, Attestation and secure device on-boarding data,

The data sharing models are further divided into **Internal** and **External** ones, concerning in the former case operations that are performed by entities/stakeholders that are placed in the same ASSURED deployment, and in the latter case any other external entity that needs to interact with an ASSURED deployment

In these flows, the types of data that are exchanged, are the following

- Operational data
- Security service data, and
- Threat intelligence data

All these data flows and the data to be exchanged should be supported by the ASSURED DLT infrastructure to allow for the correct, undisputed and privacy preserving sharing mechanism that is envisaged, to offer to the engaged parties strong integrity, confidentiality and privacy-preservation guarantees.

First, when looking at the needs of a DLT for ASSURED from a macroscopic perspective, it is essential to provide an infrastructure that can support at the same time internal and external entities and serve them with different security and data protection features relevant to their data sharing needs. As such, in ASSURED, we will make use of private and public channels within our blockchain network and follow the approach of having in general a permissioned blockchain infrastructure. As such, whenever we refer to a “public” channel, this concerns a

Funded by the Horizon 2020 Framework Programme of the European Union

channel which can be accessed by all entities of the network, provided of course that they have the right credentials to join the network.

The approach of distinguishing between private and public channels is necessary because we consider data sharing flows with both Internal and External entities to an ASSURED deployment as identified above, supporting for example in the case of our smart manufacturing demonstrator data sharing activities that are happening both within and outside the supply chain. The use of public channels will be for sharing the aforementioned data produced (within the boundaries of an ASSURED deployment) with external stakeholders, and establishing such channels will facilitate the generation of a publicly accessible data space, where important information is stored and exchanged between entities, in a privacy preserving and secure manner, as access to the public channel is subject to having the appropriate credentials to join the ASSURED DLT network, while all data exchanged there will be encrypted using ABE, meaning that only the designated recipients of the data would be able to decrypt it.

At the same time, we will use private channels for the internal parties within the ASSURED deployment. Such parties are entities that are identified based on their certificates and attributes and will make use of such channels to take part in confidential data exchange and communication. We assume that each party, e.g., a material manufacturer, within ASSURED supply chain context, will have one or more private channels, which are deployed and set up by an IT Administrator who also provides the main access requirements. It is allowed to build one whole private channel for its internal sub-parties, sub-branches, facilities, and devices communications, but also to define fine-grained levels of private channels for all its own entities. For example, having different departments within a manufacturer, could mean having different private channels set for such a department, where all entities of the same department will be able to communicate over their department's channel, and there will also be cross-channel delegates to enable those communications across different department channels. Similarly, different channels from two and more supply chain parties can communicate via the cross-channel delegates.

However, having access to the ledgers does not mean that all information to be exchanged is stored and retrieved from the ledgers. For reasons of security as well as for performance reasons, ledgers will be used to hold only limited information about the different data that is to be exchanged (like metadata, timestamps, etc), as well as pointers to where the actual data is. The actual data will be stored in an off-chain storage facility provided by the backend infrastructure of every ASSURED deployment. Furthermore, the data to be stored in this facility will be encrypted at the point of generation, by the TPMs that are operating in the devices that would like to publish data to the network, using an ABE scheme. As such, these encrypted blobs will be discoverable via the pointers stored in the ledgers, however an entity will be unable to decrypt those unless it possesses the appropriate attributes that have been selected during the step of the ABE encryption. Moreover, to allow external entities to locate data, a searchable encryption component will be offered as part of the public channel, where queries can be performed based on metadata to reveal whether a specific data is part of the system or not. In both cases, any interaction between entities and the ledger is being executed through the Security Context Broker component, which acts as a proxy, not disclosing the internal components of the ASSURED framework to external parties.

Based on all the above, in ASSURED we clearly separate how internal and external entities access the different channels and what kind of operations they can perform. As such, we allow an external entity to visit the public channels to access the public ledger to see if there is any interesting data the party would like to access. After that, such a party can request this data from the off-chain storage facility provided by an ASSURED deployment. However, it might be

Funded by the Horizon 2020 Framework Programme of the European Union

the case that there is a need to provide “regulated” access to specific external entities (such as certification bodies, law enforcement agencies, etc) to the private channels as well, mostly for auditing reasons. In such cases, these entities will be provided with the required attributes which will allow them to also access private channels.

On the other hand, Internal entities have always the ability to query the information stored in the private ledgers they can access, and thereafter discover where they can get this data from. In any case, access to the different channels will be provided by checking the attributes of each entity, based on certificates, while the actual read operations on the data would be subject to these entities possessing the appropriate cryptographic keys that will allow them to access the original content. For this reason, the overall DLT infrastructure will be protected by different security mechanisms and various cryptographic methods will be used for securing all data sharing operations over the ledgers. A core requirement for being able to operate over the ASSURED DLT network is for devices to possess a TPM, as this is central not only for the attestation operations to be performed, but also for acting as a trusted wallet that will facilitate operations relevant to the encryption and decryption of data using the novel ABE scheme proposed by ASSURED. More details on such requirements are presented in chapter 3.2.

The following table provides a detailed view of the main requirements relevant to the data sharing flows that will be performed over an ASSURED deployment and will be supported by the DLT infrastructure. These requirements (provided as DLT-SHA-xx) are extracted with regards to elaborating on the above-mentioned discussion points and taking into consideration the data flows defined in D1.4.

It is noted that details on the security, confidentiality and trust requirements that need also to be in place, are provided in the subsequent chapter 3.2.

TABLE 5: ASSURED DLT SHA REQUIREMENTS -

ASSURED DLT Data Sharing Requirements (DLT-SHA)
DLT-SHA-01 - The DLT infrastructure shall be constructed in such a manner that information can be isolated between different entities by employing different private channels
DLT-SHA-02 - The DLT infrastructure shall provide a public channel to allow data sharing with external entities
DLT-SHA-03 - Access to the DLT Infrastructure shall be based on certificates owned by the different entities and resolved via an ABAC layer as the ASSURED DLT network is a permissioned blockchain infrastructure
DLT-SHA-04 - An IT Administrator of the organisation shall create and define the access policy
DLT-SHA-05 - Policies shall be deployed as smart contracts over the DLT Infrastructure to allow for automated policy checking and compliance enforcement
DLT-SHA-06 - Users of the same department of an organisation within an ASSURED deployment should be placed in the same private channels
DLT-SHA-07 - Users of different departments within an organisation should have access to the public ledger
DLT-SHA-08 - Users external to an organisation shall have access to the public ledger only
DLT-SHA-09 - Users that want to interact with the ASSURED ecosystem shall have a TPM in their devices
DLT-SHA-10 - The trusted blockchain wallet of each user shall be hosted by the TPM owned by the user

Funded by the Horizon 2020 Framework Programme of the European Union

DLT-SHA-11 - Users external to an organisation shall interact only with the Security Context Broker
DLT-SHA-12 - The data to be exchanged over ASSURED shall comply with a specific data model and fall within the 3 different data types identified.
DLT-SHA-13 - The data coming out of the devices shall be transformed into input readable by smart contracts via an API Layer
DLT-SHA-14 - The data coming out of the devices shall be encrypted using ABE
DLT-SHA-15 - Data generated from the various entities shall be stored in an off-chain storage
DLT-SHA-16 - Metadata relevant of the generated data shall be stored in the ledgers
DLT-SHA-17 - The storage location of stored data shall be kept on the ledger using a pointer
DLT-SHA-18 - Users shall be able to search over the encrypted data through querying metadata via the Searchable Encryption component
DLT-SHA-19 - Users shall download assets from the off-chain storage by providing the necessary pointer to the Security Context Broker

4.2 DLT SECURITY, CONFIDENTIALITY AND TRUST REQUIREMENTS

This chapter elaborates on the security and trust guarantee data sharing requirements identified in chapter 2 of the ASSURED deliverable D1.4 “Report on Security, Privacy and Accountability Models” and extracts out of those the ones relevant to the ASSURED DLT infrastructure. In this context, the following lines start with grouping the requirement based on the high-level security requirements identified in D1.4 that are relevant to operations that can be supported by DLTs. Thereafter these are fleshed down into DLT relevant requirements (provided as DLT-SCT-xx) taking into consideration the main components that are part of a DLT network, such as peers, access

TABLE 6: ASSURED DLT SCT REQUIREMENTS - DATA SECURITY AND INTEGRITY

SR1 Data Confidentiality and Integrity *Data must be protected with appropriate controls to ensure their integrity, confidentiality, and availability throughout its entire life cycle*
ASSURED DLT SCT Requirements
DLT-SCT-01 - The deployment of the DLT infrastructure shall include the necessary number of peers to guarantee the integrity of the data stored in the ledger
DLT-SCT-02 - The consensus mechanism to be used shall provide the means to guarantee the integrity of the data stored in the ledger
DLT-SCT-03 - The DLT infrastructure shall provide the necessary availability guarantees for the data by utilising an off-chain data storage facility
DLT-SCT-04 - The data relevant to different operations shall be recorded on the ASSURED DLT Infrastructure in a performant and trusted manner through the smart contracts
DLT-SCT-05 - The original and full copy of a given encrypted data shall be stored on the off-chain storage to allow the ledgers to be performant and meet availability SLAs
DTL-DEC-06 - Data stored on the ledgers shall enable confidentiality by being encrypted
DLT-SCT-07 - The encrypted data stored on the ledgers shall be a pointer to the actual data in the off-chain data storage
DLT-SCT-08 - The off-chain data storage shall be accessible only via the Security Context Broker, with the latter acting as a proxy between the off-chain data storage and the requestor

Funded by the Horizon 2020 Framework Programme of the European Union

DLT-SCT-09 - Data sharing shall be proceeded via the use of smart contract, and the data sharing preference and policy should be embedded into the contract
DLT-SCT-10 - The authentication attributes of a devices shall be stored in its TPM Wallet
DLT-SCT-11 – The Security Context Broker shall correctly authenticate itself with the backend, and there will be a secure communication channel between them during data storage and retrieval in and out from the backend
DLT-SCT-12 – The Security Context Broker shall communicate with the backend over a secure communication channel
DLT-SCT-13 – The Security Context Broker shall securely obtain the storage pointer of an attestation report and pass it to the private network.
DLT-SCT-14 – The Security Context Broker shall securely obtain the storage pointer of an attestation report, and pass it to the Searchable Encryption component
DLT-SCT-15 - The Security Context Broker shall play the role of the access administrator, operating the ABAC layer.
DLT-SCT-16 - The Security Context Broker shall interact with the chaincode to monitor the policy control processes.
DLT-SCT-17 - The Security Context Broker shall produce the metadata for the attestation assets to be encrypted by the Searchable Encryption Component

TABLE 7: ASSURED DLT SCT REQUIREMENTS - AUTHORISATION AND ACCESS CONTROL

SR2 Authorisation and Access Control *The participating users, devices and stakeholders should act according to the security and privacy policies as dictated by the data sharing preferences (deployed via smart contracts) based on the sensitivity of the operational and/or security related data to be exchanged between either internal or external members of the target supply chain ecosystem. Thus, a specific dataset can only be read by users matching such pre-defined access policies (preferences) based on their shown (verified) credentials – to be protected from eavesdropping/leakage. In case such policies need to be updated, during runtime (e.g., specification of different attributes for accessing specific system raw (attestation) data), this should be reflected through the deployment of new smart contracts*
ASSURED DLT SCT Requirements
DLT-SCT-18 - The access policies to the channels shall be provided by the ASSURED Policy Framework
DLT-SCT-19 - The access policies to the channels shall be included in the smart contracts through the Smart Contract Composition Engine
DLT-SCT-20 - The access policies to the channels are deployed as chaincode on the different private and public peers by the Smart Contract Composition Engine
DLT-SCT-21 - The ABAC layer shall be executed as part of the functionality to be offered by the Security Context Broker
DLT-SCT-22 - Data stored in the ASSURED DLT Infrastructure can only be read by entities that possess the appropriate attributes
DLT-SCT-23 - The ABAC shall communicate with a trusted Authentication Service
DLT-SCT-24 - The ABAC shall verify via the credentials of an entity that have been obtained by the Blockchain CA and the policies defined by policy engine.
DLT-SCT-25 - The ABAC shall provide the different permissions to the parties that want to access the DLT infrastructure
DLT-SCT-26 - The ABAC shall verify via the credentials of an entity that have been obtained by the Blockchain CA and the policies defined by policy engine.

Funded by the Horizon 2020 Framework Programme of the European Union

DLT-SCT-27 – The DLT Infrastructure shall make use of a Blockchain CA to issue credentials relevant to the access to the blockchain network
DLT-SCT-28 – The DLT Infrastructure shall make use of a privacy CA to issue credentials relevant to the devices TPMs
DLT-SCT-29 – The Blockchain CA shall be used to manage and verify the credentials of the devices

TABLE 8: ASSURED DLT SCT REQUIREMENTS - CRYPTOGRAPHY

SR3 Cryptography *Having strong cryptographic primitives is a fundamental requirement of any security-oriented system. What is needed towards this direction is a good source of entropy that will be utilized in a secure pseudo-random number generator (PRNG) so that the keys generated by the system are secure. To make good use of this source of entropy, we also must ensure that the cryptographic primitives deployed in a root of trust and related systems are fit for purpose*
ASSURED DLT SCT Requirements
DLT-SCT-30 - The data to be stored on the ASSURED DLT Infrastructure shall be protected by cryptographic primitives coming out of the TPM of the devices that also perform the attestation
DLT-SCT-31 - The different cryptographic keys to be used by devices shall be only stored in the TPMs of the devices
DLT-SCT-32 - The devices that communicate with the ASSURE DLT shall securely store in their TPMs the secret key material and prevent leakage of this information using trusted cryptographic protocols.
DLT-SCT-33 – The TPM shall store the main secret key securely,

TABLE 9: ASSURED DLT SCT REQUIREMENTS - DATA ENCRYPTION

SR9 Data Encryption *A party is allowed to use an encryption algorithm, e.g., Attribute-based Encryption (ABE), to encrypt a piece of information data under various attributes and policies (as depicted by the data sharing preferences – cf. SR2) to output a ciphertext, so that only data seekers exhibiting valid attributes and credentials can decrypt and reveal the underlying plaintext*
ASSURED DLT SCT Requirements
DLT-SCT-34 - The encryption methods to be utilized by the TPMs shall utilize an ABE scheme to allow data to be read (decrypted) by entities that possess the correct attributes
DLT-SCT-35 - Decryption of data fetched from the ASSURED DLT Infrastructure shall happen at the device level of the data requestor using its attributes and keys stored in the TPM
DLT-SCT-36 - The encrypted data to be stored in the ASSURED DLT Infrastructure shall be made searchable in its encrypted state, using Searchable Encryptions
DLT-SCT-37 - The DLT infrastructure shall provide the necessary availability guarantees for the data, by employing an off-chain storage facility
DLT-SCT-38 - The Searchable Encryption Functions shall be provided at the public peer level
DLT-SCT-39 - Queries over the searchable data shall be performed via the Security Context Broker acting as a proxy between the requestor and the Searchable Encryption pool
DLT-SCT-40 - The ABE component shall enable devices to encrypt attestation reports

Funded by the Horizon 2020 Framework Programme of the European Union

DLT-SCT-41 - The ABE component shall enable devices to perform the corresponding decryption with the help of TPM
DLT-SCT-42 - The encryption performed shall be based on attribute and policy belonging to the TPM devices
DLT-SCT-43 - The decryption key shall be protected and issued by the TPM based on attributes provided by the device
DLT-SCT-44 – The TPM shall use the main secret key to generate an attribute-based key that helps devices to perform valid decryption.
DLT-SCT-45 - The Searchable Encryption component shall generate a keyword index structure and further encrypt the structure along with the data
DLT-SCT-46 – The secret key of Searchable Encryption component shall be hosted and protected by the Security Context Broker's TPM
DLT-SCT-47 - The Searchable Encryption component shall return replies to queries performed by users via the Security Context Broker

TABLE 10: ASSURED DLT SCT REQUIREMENTS - TRUSTWORTHINESS OF EXCHANGE DATA

SR10 Trustworthiness of Exchanged data *The data sender/receiver/endpoint are authenticated with secure identities check based on (verifiable) credentials linked to valid attributed required for accessing specific information data (cf. SR4). Furthermore, data's confidentiality is protected via encryption*
ASSURED DLT SCT Requirements
DLT-SCT-48 - The devices that want to access the ASSURED DLT Infrastructure shall be granted the relevant certificated by the ASSURED Blockchain CA
DLT-SCT-49 – The privacy CA shall be used to manage and verify the credentials of the TPMs

TABLE 11: ASSURED DLT SCT REQUIREMENTS - NON-REPUDIATION AND ACCOUNTABILITY OF ACTIONS

SR13 Non-repudiation and Accountability of Actions *Actions should be non-repudiable and all system entities should be held accountable of their actions. For instance, a data subject cannot refuse the authorship of an attestation report that has been shared on the ledger for verification from other users/devices*
ASSURED DLT SCT Requirements
DLT-SCT-50 - The attestation data stored on the ledger shall not be able to be removed or amended once the initial transaction has been executed
DLT-SCT-51 - The hash value of the plaintext version of an attestation function shall be stored on ASSURED ledger for integrity check-up later
DLT-SCT-52 - The data sharing event and status shall be recorded on the ledger for later auditing purposes.
DLT-SCT-53 - TPMs shall be deployed in peers to guarantee their correct operation
DLT-SCT-54 - The attestation data stored on the ledger shall not be altered by any action/user
DLT-SCT-55 - The data stored on the ledger shall be validated by a set of peers during the transaction
DLT-SCT-56 - The consensus mechanism to be used shall take into consideration the resolutions of the majority

Funded by the Horizon 2020 Framework Programme of the European Union

5 SMART CONTRACTS IN ASSURED

5.1 TYPES OF SMART CONTRACTS IN ASSURED

Based on the general architecture of the ASSURED blockchain environment (Figure 10) and the overall sequence of actions described in Chapter 2.2, the ASSURED DLT component essentially acts as the **"*man in the middle*" between edge devices** (requested to provide verifiable evidence on their level of assurance and trustworthiness by executing various attestation schemes [51]) and **external stakeholders** that may want to access such attestation-related data for certifying the security level of the overall SoS-enabled ecosystem. As such, among other things, the role of the platform can also be generally thought of being that of a broker, who is doing the matchmaking between the transacting parties. However, as also aforementioned, another core innovation of ASSURED is the enforcement of the compiled optimal set of attestation policies through the use of smart contracts mediated by the Security Context Broker (SCB).

Therefore, the architecture includes two types of **Smart Contracts (SCs)** to be implemented as part of the workflow. Since the smart contracts are pieces of code, there should be a definition of the functions they will provide to the rest of the components [2]. The first step for that design is the description of those smart contracts and the interactions with other components in the general architecture, as will be documented in the remained of this chapter. *Note that we focus on detailing the process for managing the deployment, execution, recording and sharing of the attestation smart contracts since this is a rather complex process*. Defining the smart contracts for querying specific attestation results is more straightforward and the focus there would be on defining appropriate access control policies to be enforced by the Security Context Broker (SCB) (Chapter 6.2.7) and the Membership Service Provider (MSP) (Chapter 6.2.8).

In what follows, we describe in more details the main concepts surrounding the operations of the two types of required smart contracts. This sets the baseline of the envisioned functionalities of smart contracts, as a **core enabler for attestation execution and (secure ad privacy-preserving) data sharing** that will be further modelled in D2.5 [55].

Smart Contract depicting the scheduling attestation policies	
Objective	SC that needs to provide all the necessary functions and information for a set of edge devices to execute a remote attestation process based on the attestation schemes described in D3.2 [51]. This SC will be based on the scheduling policies as outputted by the Policy Recommendation Engine [2] for capturing the type of attestation schemes need to be executed by all edge devices towards achieving the required level of trustworthiness.
Triggered by Who	This is triggered (and monitored) by the Security Context Broker upon reception of the attestation policies by the Policy Recommendation Engine.
Stakeholders & Components involved	**Policy Recommendation Engine** for providing the optimal set of policies to be deployed as smart contracts; **Security Context Broker** and **Smart Contract Composition Engine** for creating the appropriate smart contract logic based on the received attestation policies; **Edge devices** for which these attestation policies are destined and that need to read and execution the attestation policy prior to recording the

Funded by the Horizon 2020 Framework Programme of the European Union

	attestation result on the ledger.
Input	Attestation policy written in an MSPL format [2]. This contains all the necessary information including: ✓ Types of attestation tasks to be executed per device; ✓ IDs of the devices – acting as prover and Verifier – to take place in the attestation process; ✓ Order of execution of both the attestation and operational tasks running in the target device; ✓ Execution time to be allocated per task.
Outcome	A transaction stored on the ledger including the result of an executed attestation process. Recall that this is binary outcome representing whether the attested device is at a correct state or not. The accompanying system traces, based on which the attestation was performed, are stored in the off-chain database (in an encrypted format) and a pointer is added in the attestation report block.
Smart Contract depicting the querying of attestation related from external stakeholders	
Objective	SC that needs to provide the necessary access control policies for checking the types of attributes and privileges to be exhibited by a requesting entity wishing to query some of the recorded attestation related data.
Triggered by Who	This is triggered by an external stakeholder that wishes to query for some attestation data of interest based on the metadata it has identified on the public ledger. In this case, it contacts the SCB that initiates the appropriate ABAC mechanism to be executed.
Stakeholders & Components involved	**Security Context Broker** for providing the appropriate access control policies to be depicted through the smart contracts. The SCB also initiates the ABAC process to be executed; **External Stakeholders** wishing to query for some attestation related data of interest. First, they access the public ledger where metadata (regarding the attestation data stored in the private ledgers) can be identified. If the external stakeholders identify something of interest, then they might request to get access to the attestation report and the accompanying system traces.
Input	Access control policies depicting the list of attributes that the requesting stakeholders need to exhibit.
Outcome	A transaction stored on the public ledger including the result of this data querying operation. Essentially which type of attestation report (ID of the proving device) was shared with whom external stakeholder.

5.1.1 Smart Contract Entities and their Roles

In this chapter, we will introduce the entities related to ASSURED smart contract (SC) and their respective roles. This can be summarised in the following table.

TABLE 12: ENTITIES RELEVANT TO ASSURED SMART CONTRACTS

Name	Role
Devices in the same channel	Entities that would like to access smart contracts for the execution of attestation and for new enrolment verification.
API	In general, it helps devices to access and communicate with smart

Funded by the Horizon 2020 Framework Programme of the European Union

component (for smart contract)	contracts. Specifically, the API component is used to interact with the TPM-based Wallet for each device to execute the functions of the smart contract defined in D2.2 and to be further elaborated in D2.5.
Chaincode component	This entity takes charge of (i) execution of smart contracts; (ii) taking input to and yielding outputs from smart contracts; (iii) interacting with channel ledger for status update. The chaincode, in ASSURED, that means the programming scripts of smart contracts that are executable and readable on ledgers. It will be transferred via the form of transaction to peers who will install it on ledger, and then later be invoked by devices.
Security Context Broker (SCB)	SCB will interact with smart contracts, so that it can control (1) which contracts could be accessed by which devices; (2) based on smart contracts design and the policies, maintain the ABAC - controlling who can access to the channels; (3) and the SCB will be also responsible for circulating (and enforcing) the secure enrolment policy, in which prior to a device being allowed to join the blockchain network, it needs to prove that the attribute key is created correctly based on a policy that represents the correct configuration the device must have.
SC Initializer and Deployer	Once a smart contract is deployed on a ledger, the initializer is engaged and calls for its execution, via the form of transaction. Further, the deployer is mainly used to deploy smart contracts on ledger. In ASSURED, this is covered by the chaincode component.
Ledger	The final outputs of the smart contracts will be updated to the channel's ledger at the end.
Endorser	This role is taken by the ASSURED blockchain network's peers. An endorser will validate a given transaction with smart contract initialisation/call, and further execute the contract. Then it will send the results as responses back to the devices.
Orderer	Given the smart contract execution responses (in the form of transactions) - which are sent by devices, orderers will order and put them into a block.
SC function	Within a smart contract, there are several predefined functions (please refer to the D2.2 - MSPL low level translation to smart contracts). These functions mainly reflect policy enforcement and the corresponding operations, for example, given a policy condition, if that is matched, some actions will be triggered.

5.2 SMART CONTRACTS WORKFLOWS

5.2.1 A General Description

In the context of ASSURED, a smart contract works as follows.

Much like a normal smart contract used in other blockchain platforms, an ASSURED smart contract will capture and reflect business logics. These logics will be seen as policies and recommended by the policy recommendation engine. Taking the policy as input, the smart contract composition engine interacted with the security context broker will convert the policy into a series of operations, which are described as smart contract functions in ASSURED deliverable D2.2 [2].

Funded by the Horizon 2020 Framework Programme of the European Union

After that, we will have the programmable logic scripts of the smart contract. But these scripts may not be directly used for the ASSURED blockchain platform. We will need to compile them into the chaincode - a further form of its programming format which can be read and understanded by the blockchain peer.

Then, the SCB can send the chaincode as a transaction “TX”, to the blockchain peer. In this type of transaction, input is the chaincode, and the output is null. Before the above “Send stage”, the API component will compose and sign the transaction, e.g., via secp256k1.

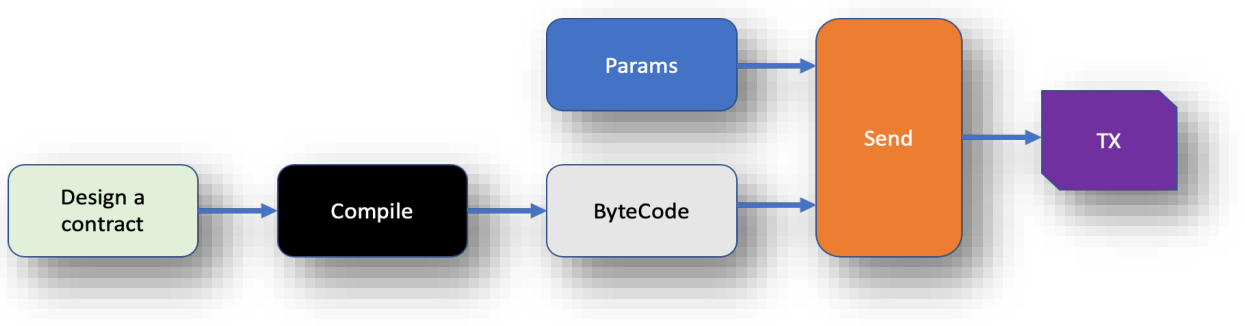


FIGURE 3: A GENERAL CONVERSION OF SMART CONTRACT

Take a further closer look at the smart contract workflow. We can see that: there exists a ledger user with its account (public key as account address and its own private key, e.g, using for signature). This user may write programming logics, e.g., via JavaScript, Solidity, and further compile these logics, e.g., via online compiler EthFiddle[45], into chaincode.

For example, for a Solidity-based program (which is a programming code written based on Solidity) to output “Hello” could be like the following example showcased in Figure 4 and the chaincode that could be understood by the ledger peer may look like the example provided in Annex A.1.

```
pragma solidity 0.4.24;
contract Greeter {
function greet() public constant returns (string) {
        return "Hello";
        }
}
```

FIGURE 4: A SOLIDITY EXAMPLE

[45] https://ethfiddle.com/

Funded by the Horizon 2020 Framework Programme of the European Union

After the conversion into chaincode, the user further makes use of the API component to sign the message while providing the input of chaincode and broadcasts the message and its signature as a transaction. This can be seen as a method call on a smart contract that is committed via the transaction. The transaction will finally be sent to the transaction pool, waiting for being mined on the ledger.

FIGURE 5: TURN SMART CONTRACT INTO TRANSACTION

A ledger peer node, within the same network or channel (where the user stays at), now is able to pick up this transaction, and check if this transaction only has “from” but no “to”, meaning this transaction is only sent by the user shown via “from” tag but not sent “to others”. This indicates this transaction’s input is a smart contract chaincode. The peer node then merges the chaincode into a block, puts the block on ledger, and updates the whole status of the ledger (to others within the network, including other nodes).

Now the smart contract is merged and appears on the ledger, and it has its own account - smart contract account - that has a unique address and other information. After that, if a user would like to execute the smart contract, it will send an execution request with the help of the API or so-called blockchain client API - converting the request into a transaction format and further sending the transaction to a peer node (now the peer is known as the endorser). This is shown in Figure 6. The peer is then able to execute the smart contract, and then wrap the execution result into a transaction package which will be sent to an orderer. The orderer will order all received transactions and further pack them into a block. The block will be returned to the peer which will further check the ordering and the validation of the transactions within the block one more time. And finally, the block will be merged on the ledger as a new block for the ledger held by the peer.

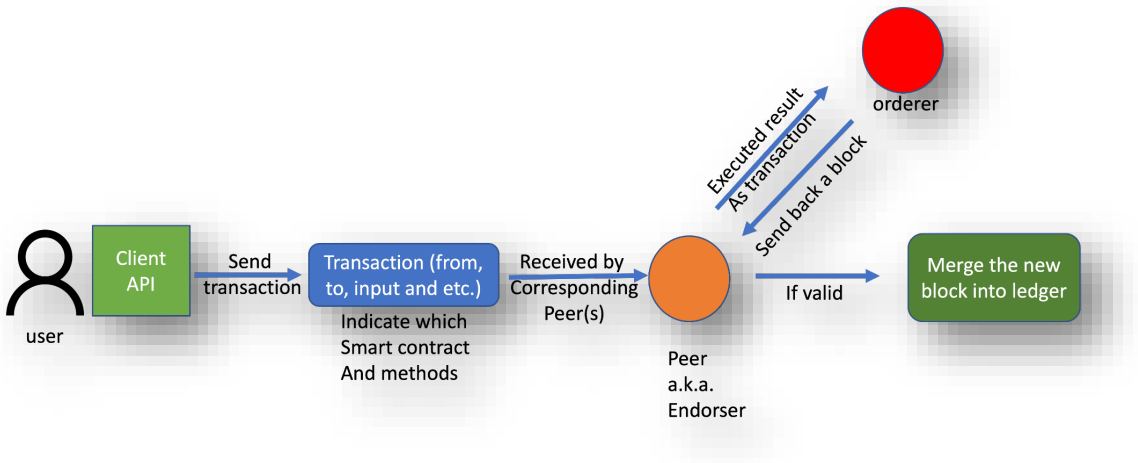


FIGURE 6: THE EXECUTION OF SMART CONTRACT

Funded by the Horizon 2020 Framework Programme of the European Union

5.2.2 Specific Setting in ASSURED

In the ASSURED DLT framework, we are going to deploy attestation within smart contracts to enable attestation to be supported by ASSURED DLT.

To this end, we mainly follow the general design, deployment and execution of the smart contract which is mentioned previously in our project. In general, we will make use of a policy recommendation engine and smart contract composition engine to obtain the policies that are needed for the attestation reports and the related data sharing, and then further input the policies into the composition engine that guides an entity, e.g., the security context broker, to design the smart contract. Recall that in the ASSURED deliverable D2.2, we have defined the smart contract functions. There are online and offline procedures. For the former, we provide functions for nonce generation, attestation verification, validation for report signature, endorsement policy design and update, attribute obtain and offload memory. And we put the preparation for verification, validation policy generation, encryption and decryption functions in the offline stage.

From the descriptions above, we may see that the smart contract functions are mainly designed to support attestation related operations, policy generation and update, and data encryption and decryption.

1. ***Deployment.*** The next figure depicts the initial deployment of the attestation-based smart contract.

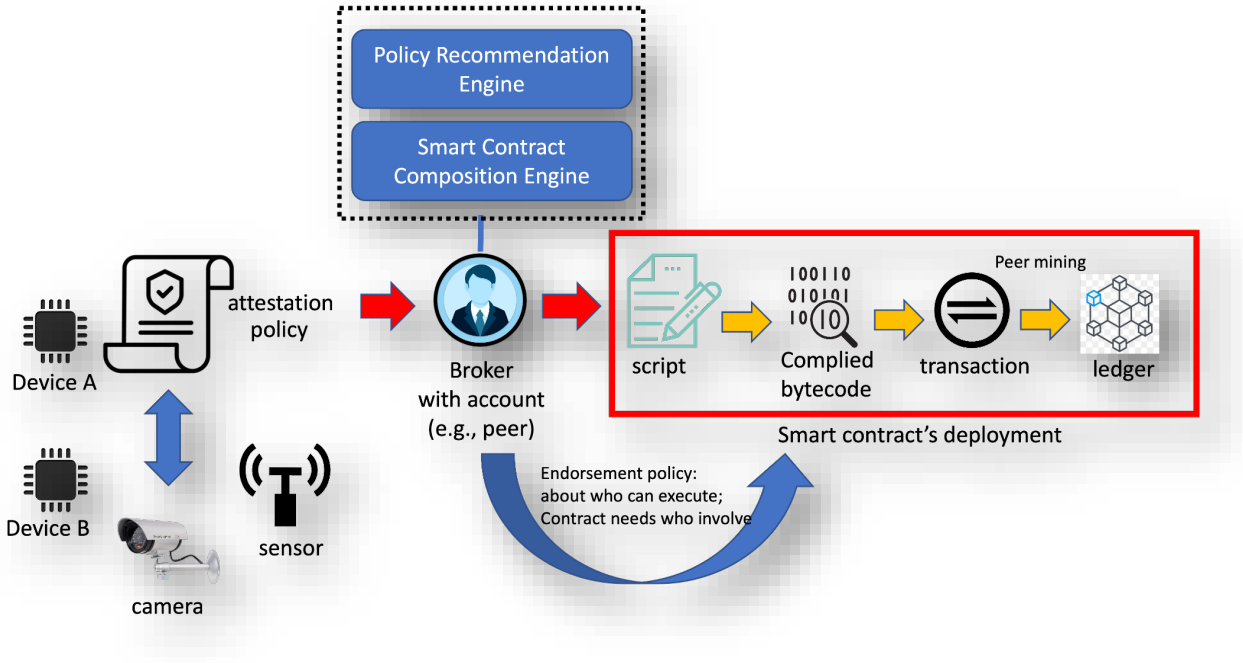


FIGURE 7: ATTESTATION BASED SMART CONTRACT DEPLOYMENT

We assume that the devices and the SCB above are registered within the blockchain network with their respective credentials, and the credentials are granted by the Blockchain CA. And they are all able to access blockchain API interfaces. We convert the attestation policies for various devices into programming logics with the help of the composition engine. This is guided and led by the security context broker (SCB) who intakes the policies from the policy

Funded by the Horizon 2020 Framework Programme of the European Union

recommendation engine. After that, the SCB is easy to propose a script, which is based on programming scripts, say Solidity or JavaScript. The script will be further converted into a chaincode which can be read and understood by blockchain peers. This part is with the help of smart contract composition engine. And the SCB will send the chaincode as a transaction, and the transaction will be further sent to the peers within the channel. Since the peers notice the transaction is already there, then they can further create events to notify the edge devices of new policies deployed. We note that this event creation can be monitored by the SCB, just in case the peers fail or ignore to do so. And the endorsement policy of this smart contract will be pre-defined by the SCB, so that one (e.g., a channel user, or ledger client) can clearly understand that this smart contract requires who (in the same channel) to execute the contract and later have corresponding validation. Much like the general description, the transaction will be batched with others into a block during the ordering service. And at last, a peer node within the channel will merge the new block (with the chaincode) on the channel ledger. Note that the chaincode of the contract, hash of the chaincode and its unique ID will be shown on the ledger, within this merged block. Note that in the ASSURED project, we will implement the attestation verification algorithm of the attestation within the smart contract and meanwhile, other functionalities like encryption/decryption, endorsement policy setting, will be also included in the smart contract as methods/functions.

2. ***Attestation Challenge Generation.*** After the chaincode is deployed on the ledger, a device is able to use it for attestation.

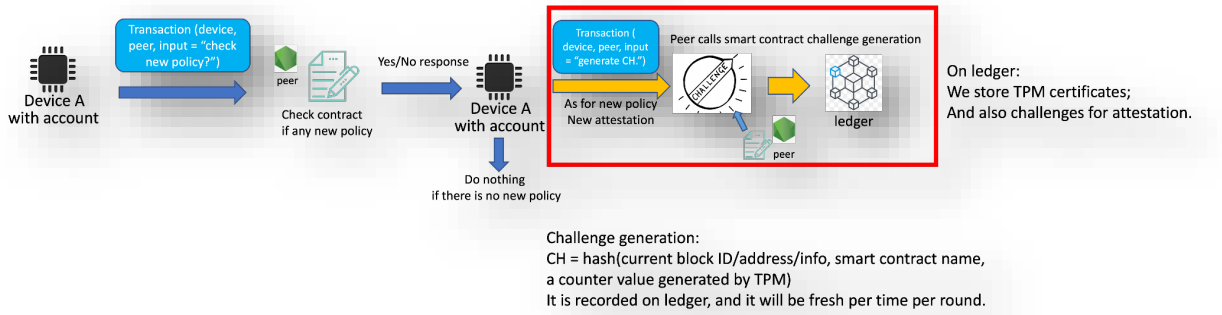


FIGURE 8: ATTESTATION SMART CONTRACT CHALLENGE GENERATION

In the context of ASSURED, we will enable a device acting as a verifier to read the attestation contract and further initiate the attestation process for the designated prover. Specifically, a device B (as a verifier) can first check if there is a new policy there that requires a device A (as a prover) to perform a new attestation. To check this, device B is allowed to request the peer to search if there are any new policies via smart contract. If there is a new policy, then the device B will initiate a new attestation process for A.

Before performing the attestation, the device A needs a challenge for the attestation. To do so, the device needs to send out a message as a transaction requiring an attestation challenge via the challenge generation algorithm in the smart contract. By receiving this message (as a transaction), the peer can locate the type of the contract for the device, and then executes the challenge generation algorithm to yield the challenge. The challenge will be further sent back to the device because of smart contract execution.

Once the device reads the challenge from the reply, it can execute the attestation algorithm (for the verifier) to output a valid attestation. In the above process, the challenge CH is a hash value of the current block ID/address/info, and the name of smart contract, and a counter value

Funded by the Horizon 2020
Framework Programme of the European Union

generated by TPM. We note that the challenge will be regarded as a part of the attestation log file that will be further stored on the ledger later.

3. ***Verification of Attestation.*** A device, say A, now can execute the attestation algorithm with its current status and the challenge.

Then we will need a verifier for the resulting attestation. During the creation of the attestation policies, trust-aware service graph chains have been generated. And in the chains there has been identified who essentially can act as the verifier for each device (attestation prover), for example, the verifier for this device's attestation is called device B. The device A will send out the results as an attestation report (with its digital signature on top) to device B, in which the attestation is signed by the TPM AK of the prover A and will be also signed by the verifier B as well as the internal private key of the verifier B which was created when registering to the blockchain network. The device B then will act as a verifier for the attestation, and it will send out its verification results along with the attestation results as a transaction.

After that, device B is allowed to send the results of the attestation (along with the attestation proof given by device A) to the peer as a transaction, so that the peer can merge the pack of the information into ledger.

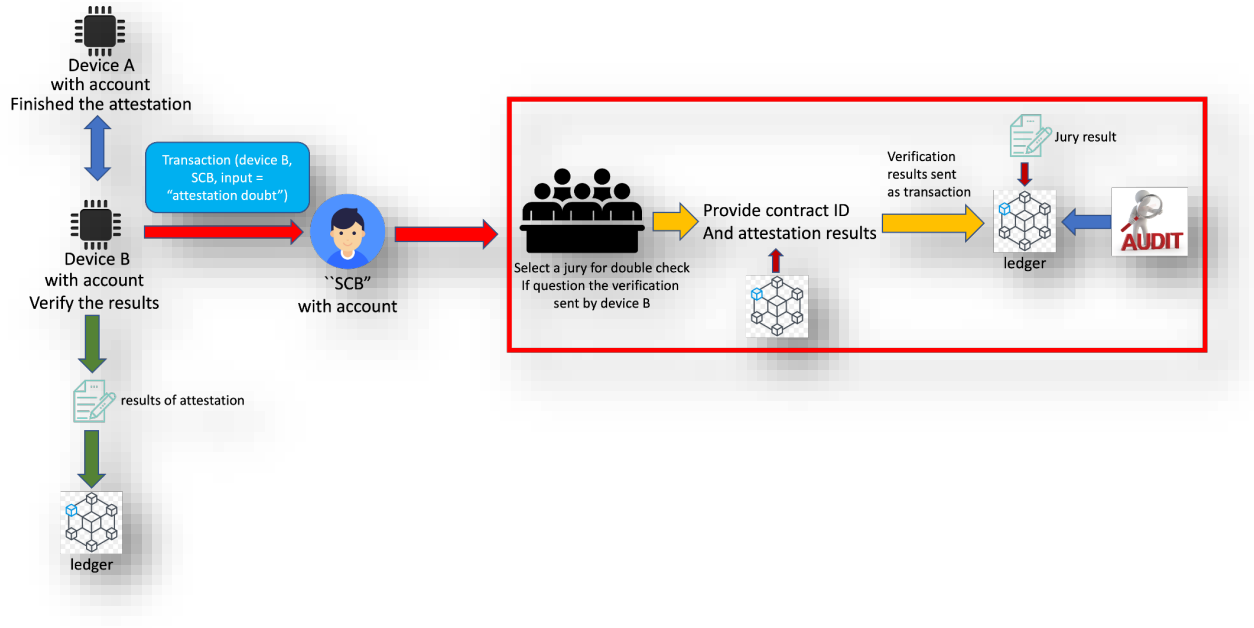


FIGURE 9: ATTESTATION SMART CONTRACT VERIFICATION.

The attestation data is shown on the ledger, and if there is a disagreement on the verification between the two devices, for example, device A does not agree with device B's verification results. Then device B will send a re-verification request to the SCB. The SCB then gets notification, and it randomly selects a group of Jury for verification of the results to check if there is any mistake or doubt on the verification. The Jury-based Attestation process then is initiated. From the previous transaction (with verification results) sent by device B, which is stored on ledger, every jury member is able to read the results from the ledger and the contract ID will be provided by the SCB. Each of the jury members is required to execute the verification with the provided results, and then all of them should make a consensus to vote if the results are valid and they accept the results - i.e., if the attestation and the verification are valid. The voting result will be sent out as a transaction and merged on the ledger as well. From the above, the inputs and re-verification results are also stored on the ledger, which makes future

Funded by the Horizon 2020
Framework Programme of the European Union

auditing (say by external third party) feasible via the call of smart contract with the stored results. For the above jury-based verifying approach, the size of the jury group is decided by the SCB.

We state that the above attestation can be further revised for new devices joining into the network. A new device here may be required to perform the configuration state check to the SCB, in which the attestation smart contract can be retrieved from the ledger with the help of the SCB. And then the SCB will check the configuration result. If that is correct, then the SCB will enable the new device to join the network. Here, the secure enrolment policy needs to be executed by the SCB for the newly joined device. And the device's configuration state can be checked by executing the configuration integrity verification protocol. If the SCB allows the new device to join, the Blockchain CA will be notified to grant a credential for the device. And before that the device must use its TPM to verify its attributes to a privacy CA, so that it can obtain a token for the Blockchain CA's credential generation. From now on, the device's in and out of the channel will depend on the check on the credential via the ABAC service (supported by the channel peers).

Funded by the Horizon 2020
Framework Programme of the European Union

6 THE ASSURED BLOCKCHAIN ARCHITECTURE

6.1 ARCHITECTURE OVERVIEW

As identified in previous deliverables, and stemming out of the overall ASSURED conceptual architecture [54], the role of the blockchain in the frame of the project has to do with **recording and sharing essential information relevant to the execution and output of the optimal set of attestation policies enforced**, to allow different systems to work in a smooth manner, letting their internal entities interoperate in a trusted and secure manner, allow **cross-department interoperability and information sharing**, as well as expose certain parts of information to the external world.

However, besides only security, **privacy** is also considered one of the core requirements that must be managed efficiently together with **scalability, smart contract verification, data storage, consensus mechanisms, etc**. Taking into consideration that most users and devices should be disenfranchised from this process since it cannot be expected that they will have a clear understanding of the various data security and user privacy implications, it is imperative to build **new on- and off-chain data management models and services** of privacy and data protection and the technologies that encode them.

In this direction, ASSURED enables enhanced **data security, integrity privacy and ownership safeguarding** (security- and privacy-by-design) and **data provenance and sovereignty** checking mechanisms. The platform uses blockchain-based distributed ledgers for offering enhanced data and transaction security. To this end, ASSURED protects operational and attestation-related data and resources against leak or improper modifications, while at the same time ensures data availability to legitimate users and devices. Internal storage and ledger infrastructures, handling attestation related data and monitored system traces, can track its provenance and are regularly audited to comply with specified security and privacy policies and regulations. This way devices are in control of their own privacy and that of their data. For the former, privacy requirements are described through privacy-related policies [53] where the type of crypto primitives to be used by edge devices are described towards achieving the necessary properties of *anonymity, unlinkability, unlinkability and unobservability*. These policies are afterwards translated in the appropriate smart contracts, following the principle of user privacy empowerment. Depending on the required privacy level, privacy enhancement is achieved through the use of the TPM-based Wallets as a central building block towards the provision of privacy-preserving signature schemes (Direct Anonymous Attestation (DAA)) [51]. By assuring auditable, security and privacy policy compliant actions, ASSURED also guarantees that application ecosystems where such policies have been technically enforced are highlighted.

In this context, the overall blockchain architecture to be devised has to support the following principles based on the different stakeholder and system requirements already identified in the context of the ASSURED project:

- A **permissioned network** should be established, where members of the network are authenticated and authorised users, thus, the overall network shall not be open, to provide the maximum-security guarantees to the engaged stakeholders. For this manner two types of certificates will be used, one for having access to the blockchain network and one to govern the privacy and security keys of a device (in its TPM-based Wallet). As described in Chapter 2.2, the latter is for supporting the verification of the attributes

Funded by the Horizon 2020 Framework Programme of the European Union

presented by a device (during secure enrolment to the Privacy CA). If this is successful, then a token is credited to the device that can then be forwarded to the SCB for checking prior to the Blockchain CA creating the appropriate certificates for the authentication and authorization of the requesting user.

- Each organisation participating in the network shall operate its own **private channels** (private ledgers), to isolate data and operations from the rest other participants of the network. As such, devices belonging and operating within the boundaries of a given organisation do communicate with each other over private channels which are designed in such a manner to conceptually cover specific “departments” or “units” of an organisation, and thus isolating the operations of each such department/unit. For instance, consider the Human Robot Safe Interaction scenario in the envisioned Smart Manufacturing use case: All devices belonging to a specific manufacturing floor (governed by the Industrial Gateway) would belong to the same private channel so that they can immediately share information regarding the operational assurance status as needed towards the creation of trust-aware service graph chains. On the other hand, devices belong to different manufacturing floors, can again request access only if the appropriate access attributes have been granted.
- A **public channel** (public ledger) per organisation is provided, to allow cross-organisations information exchange and exposure of private channel data that can be retrieved by other entities belonging to the same organisation, and therefore allowing cross department/unit collaboration.
- Different **Blockchain Peers** are set up to support the operation of the private channels, while peers setting up the public channel are also present. Each of those peers operate the ledger and host the chaincode – essentially the act as the “*gateway*” for supporting the on-chain interactions (e.g., attestation policy query, recording, sharing) of the edge devices with the private ledgers.
- Smart contracts are deployed as chaincode to support the attestation operations between different devices and to facilitate data exchange.
- An organisation-wide **off chain data storage facility** is provided which includes the raw attestation data in encrypted format, while pointers to that data are stored in the ledger. Encryption is done in a decentralized manner by the edge devices employing their TPM-based Wallets (Chapter 6.3.2). The intuition behind this is based on the overarching theme of ASSURED towards **shifting trust from the infrastructure to the edge devices**.
- Access to all ledger’s information is based on an ABAC scheme that resolves whether an entity can have access to the blockchain network or not. This operation is managed by the Security Context Broker (SCB) together with the Membership Service Provider (MSP).
- **ABE is performed at the device level on the attestation data to be stored on the off-chain facility**, so that only devices with the correct attributes can decrypt these data. As aforementioned, all keys are managed and protected by the TPM-based Wallet offering the attributes described in the overall trust model put forth in D3.1 [56].
- A Searchable Encryption (SE) component is operating over the Public Ledger that holds encrypted metadata, allowing any entity to perform a query to identify whether certain information exists in the public ledgers and in the off-chain storage.
- The Security Context Broker is operating as an intermediate entity resolving the different requests coming from the different ASSURED components and the outside worlds and facilitates the overall communication between the peers, the off-chain storage and the other components that have to do with access control, etc.

Funded by the Horizon 2020 Framework Programme of the European Union

Following these lines, the following **Figure 10** provides an overview of the ASSURED blockchain architecture, revealing the components that are part of the **private and of the public channels**, other relevant components that operate within the boundaries of each organisation, and the components that will be deployed at the level of each device that will be using the ASSURED DLT network. As can be seen, private channels are responsible for the creation and validation of attestation smart contracts to be executed by the edge devices while public ledgers are positioned for **capturing and recording the fine-grained details of extracted metadata towards efficient attestation-related data search**. This envisaged conceptual architecture captures the following set of provided on- and off-chain control functionalities and services that will be further elaborated in Chapter 7.

ASSURED Trusted Blockchain Services: Trusted Platform Modules (TPMs) are a central building block of ASSURED secure and privacy-preserving mechanisms and form the basis for enhanced security, privacy and reliability guarantees for ledger management and maintenance. The smart integration of the TPM technology will allow ASSURED to develop new blockchain verification methods and significantly advance the state-of-the-art of blockchain operation services: (i) **secure storage:** a user can store any secrets (keys, passwords or other sensitive data) associated with a TPM, and, when authorized by the user, the TPM allows access to the user's secrets, and (ii) **secure execution:** it provides a trusted execution environment that allows the isolated, secure execution of code mainly for protecting the execution of security-relevant code.

Trusted Blockchain Wallets: These are essentially the TPM-based Wallets been hosted in each one of the edge devices. They will be used to: (i) provide strong user authentication and to securely store the user credentials based on the TPM's secure key storage, (ii) control and authorize access to private or public ledger channels based on the user authentication process (e.g., to authorize access to or operations on different ledgers), and (iii) securely and efficiently verify blockchain updates. In this way, ASSURED will significantly advance the state-of-the-art of blockchain verification methods: Unlike current mechanisms that often rely on computationally costly and wasteful proofs of work or biased proofs of stake, ASSURED will use TPMs as central building block to build a very resource-efficient and trustful two-staged blockchain verification mechanism, which will be even suitable for resource-constrained devices (such as smart devices - equipped with a TPM). From a TPM perspective, the continuous verification procedure of blockchain edits can be outlined as follows, where we will assume that all participating entities hold the current blockchain state hash inside their TPMs: In Stage 1, the device will perform a pending blockchain update, and will then determine the updated blockchain state hash based on the ledger updates and the current state hash. Then, in Stage 2, the chosen verifiers (and any other ASSURED users) are able to verify the update. This involves checking the validity of the updated blockchain state based on the block update and the current blockchain state hash. On success, the users will then replace the current state hash inside their TPMs with the updated one.

Trusted Authentication: To secure communication and prevent impersonation and man-in-the-middle attacks, peer authentication is of extreme significance. ASSURED will offer multi-tier secure authentication based on the aforementioned TPM-based Wallet trust anchor: (i) trusted identity authentication between peers, (ii) trusted membership authentication for read and write on ledger, (iii) trusted access authentication for cloud-cased storage system, and (iv) trusted actioner authentication for data search and sharing. ASSURED guarantees that a user or a party claim what it is that is exactly what it is, which means that trust can be delivered inside the physical level – providing trustworthiness for the device managed by the user.

Funded by the Horizon 2020 Framework Programme of the European Union

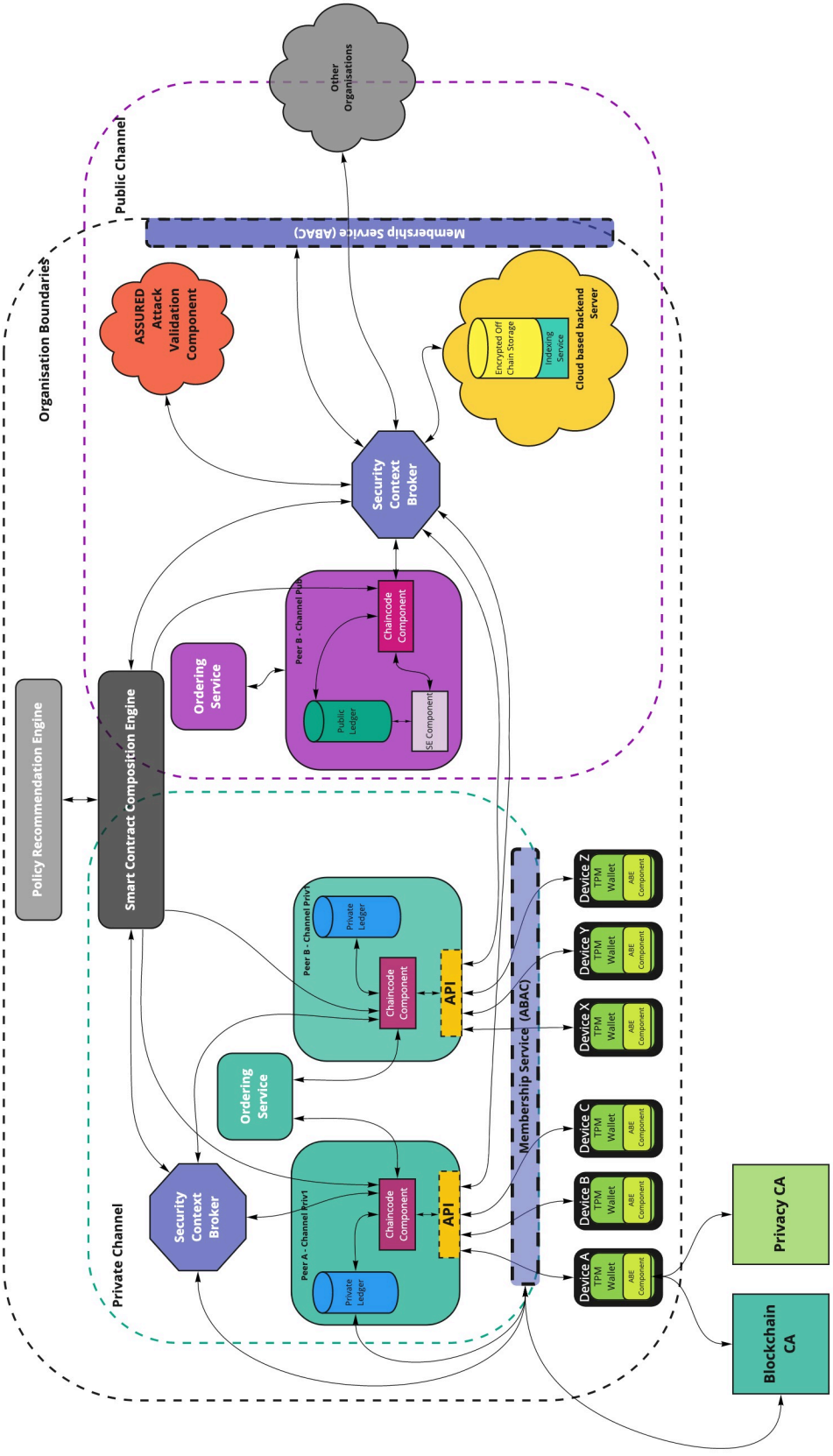


FIGURE 10: ASSURED DLT ARCHITECTURE

Funded by the Horizon 2020 Framework Programme of the European Union

6.2 ARCHITECTURAL BUILDING BLOCKS - PRIVATE AND PUBLIC CHANNELS

This chapter discusses the main components that can be found in the public and private channels of the ASSURED DLT Network. As the components that are part of each channel have an identical operation, we refrain from distinguishing between components belonging to the one or the other channel, and any differences in their operations are described within the text below.

6.2.1 DLT Blockchain Peer

A DLT **Blockchain Peer** (also called a node peer or simply peer) is the entity of the blockchain network that is hosting the chaincode to be executed (smart contract functions [2]) and the ledger where each transaction information is written. In the case of ASSURED, a DLT peer is interacting with the edge devices through their TPM-based wallet for supporting the execution of the attestation smart contract functions and the recording and sharing of the attestation reports on the ledger, as identified in the previous chapter.

Peers are considered integral parts of any blockchain network and are the main entities that perform operations and are the ones that maintain the ledgers and take part in the consensus schemes that are used to carry forward transactions and place them in the blockchain.

Relevant to their operations, peers have two main roles; they can be **endorsement peers** (e.g., executing and endorsing the transaction and sending that to the Ordering Service, without writing it on the Ledger), while all peers have the role of the **committing peers**, that validate transactions once they have been ordered and then record the transaction to the ledger. In ASSURED we do not make this distinction, as all the peers we are going to use are having chaincode installed, thus are at the same time endorsement and committing peers. Essentially, the endorsing peers are executing the transactions, they send it to the Ordering Service of the channel, and then this is received back in the context of an ordered block, which the committing peers write (commit) in the ledger.

A peer can participate in more than one private channel. In ASSURED we plan to have at least 3 different peers deployed in each organisation (for enhanced scalability), however, there will be multiple private channels support, facilitated by these peers, as the decision on the number of private channel to be used per organisation remains with the demonstrator organisation, based on their needs and strategy for dividing operations and systems (e.g. one organisation might have a private channel for each department, or this can be per types of infrastructure, per site, etc).

Moreover, a peer can be also classified as an “Anchor” peer in case it is considered as a peer that allows cross-organisation communication, deeming itself as discoverable by other peers. These peers will be the ones that will be part of the public channel, so that information can be shared both in an inter-organisation as well as cross-organisation levels. It is noted that from a deployment perspective, in ASSURED we envisage that the anchor peers will be also supporting private channels as well (e.g., operating as the same Virtual Machine), in order to provide increased network performance and refrain from adding extra peers that need to communicate with the private channels.

Requirements linked to the DLT Peer: DLT-SHA-01, DLT-SCT-25, DLT-SCT-257, DLT-SCT-50, DLT-SCT-53, DLT-SCT-55

Funded by the Horizon 2020 Framework Programme of the European Union

6.2.2 Chaincode Component

As the “Chaincode Component” in ASSURED we envisage the execution engine of the different smart contracts which are deployed within the different peers and are tasked to perform all the necessary operations to record and retrieve attestation data as it will come out of the devices that are participating in the private channels, as well as to perform operations as tasked by the Security Context Broker – essentially the creation and deployment of the necessary smart contracts that contain the attestation policies as outputted by the Policy Recommendation Engine. The Chaincode to be executed will be the one that will be deployed on the different peers by the Smart Contract Composition Engine directly in the peers, with different chaincode being present in the private and the public channels.

The input to the chaincode component will be provided by the API layer to exist in the different peers, when this regards the devices that want to push data to the network, while the Security Context Broker will be used to execute chaincode for task such as querying and retrieving operations from the ledger, etc.

Normally, chaincode runs in isolation from the other peer operations (like endorsing) making use of Docker containers.

Requirements linked to the Chaincode Component: DLT-SHA-05, DLT-SHA-16, DLT-SHA-17, DLT-SCT-04, DLT-SCT-09, DLT-SCT-16, DLT-SCT-19, DLT-SCT-20, DLT-SCT-51, DLT-SCT-52, DLT-SCT-54

6.2.3 Private and Public Ledger

The ledger is the point where information is recorded in a blockchain network and consists of the World State (a database holding the current version of the ledger’s assets) and the blockchain which provides a detailed, immutable evolution of the different assets defining how they reached the current state (which is reflected in the World State). As identified in previous chapters, in ASSURED we employ a hybrid approach where we do not store all relevant attestation data on the blockchain for performance issues as well as discoverability and sharing facilitation reasons. Instead, we make use of both the ledgers and of off-chain storage (see chapter 6.4.4), to record this data; *attestation results are stored on the ledger whereas the accompanying system traces (control-flow traces or digest lists of loaded binaries) are stored in the ASSURED Data Storage Engine*.

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 through the API layer (see below) as input to specific smart contracts, while the pointer to the off-chain storage is provided by the Security Context Broker that handles the operations of storing the encrypted attestation data to the database.

Regarding the public ledger, the information stored there are the encrypted pointers to the off-chain storage facility where attestation data is stored, and **encrypted metadata that relate to each attestation alongside with information relevant to which private channel** they originated from. This information is provided to the ledger by the Security Context Broker, in an encrypted manner, and therefore searchable encryption is used for querying over those data and discovering which private channel should be queried thereafter.

Funded by the Horizon 2020 Framework Programme of the European Union

Requirements linked to the Private and Public Ledgers: DLT-SHA-01, DLT-SHA-02, DLT-SHA-03, DLT-SHA-04, DLT-SHA-06, DLT-SHA-07, DLT-SHA-08, DLT-SHA-16, DLT-SHA-17, DTL-SEC-06, DTL-SEC-07, DLT-SCT-13, DLT-SCT-50, DLT-SCT-51, DLT-SCT-52, DLT-SCT-54

6.2.4 Searchable Encryption Component

As information to be stored in the Public Ledger will be encrypted, it is essential to provide a service that allows interested stakeholders to be able to locate whether information of their interest exists within an organisation and then point them to the right retrieval points. In order to facilitate this need, within each Public Peer ASSURED will deploy a Searchable Encryption Component 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 Security Context Broker in an encrypted manner, and this component will allow stakeholders to use a special search token corresponding to a or some keywords to perform queries over the encrypted metadata.** In case the token of the querying is valid, and the requested information exists on the ledger, this will be revealed to the asking party, reverting them to the right private channel/peers where they should query to get the full attestation report.

Requirements linked to the Searchable Encryption Component: DLT-SHA-18, DLT-SCT-01, DTL-SEC-06, DLT-SCT-14, DLT-SCT-17, DLT-SCT-36, DLT-SCT-38, DLT-SCT-39, DLT-SCT-45, DLT-SCT-47

6.2.5 API Layer

API layer provides a somewhat "*gateway*" to bridge connections and communications between devices and peers. Its main functionalities can be described into the followings:

- After the deployment on a ledger maintained by a peer (say Peer A), a smart contract is ready for invocation. A device will first send a query request through the API as a transaction so that the Peer A can receive it and then forward it to the device.
- Through the API interface, the Peer A is able to receive the execution results as responses from the device acting as the Verifier that initiates the attestation task with the prover device as dictated by the attestation contract [2].
- The API will also help the device wrap up a transaction which is to be sent to the ordering service.
- If the device would like to query and check the update status on the ledger, it will use the API to send the query transaction to Peer A, and the results will be sent back through the same interface – *assuming of course that the device has the correct privileges to read this specific attestation report*.

Requirements linked to the API Layer: DLT-SHA-12, DLT-SHA-13, DLT-SCT-04

6.2.6 Ordering Service

In the ASSURED blockchain framework, we use the concept of "permissioned" so that we always require those who want to access the ledgers to be authenticated. In this context, we will leverage a special role, called orderer, to organise the submitted transactions in order and then to form a new block.

Funded by the Horizon 2020 Framework Programme of the European Union

In the setup stage of our ASSURED blockchain, we will first define and set an organisation, and under this organisation, we set at least one orderer, meaning the orderer(s) belong to the organisation. After that, we use channel certificate authority to issue an ID credential to the orderer(s) so that their IDs are tagged with the orderer's role, and they can be authenticated within the channel.

If a device via the API layer sends a transaction to a peer, the peer will validate and execute the transaction and further return the results to the device. The peer then submits another transaction containing the results to an orderer. Assume there are many of these types of transactions sent to the orderer. It then creates blocks of transactions. Note that the number of orderers can be more than one, and they receive transactions from various peers concurrently. In this case, they must work together to turn the transactions into well-defined sequences and pack them into blocks - which are the blocks of the ledger. Since there could be more than one orderers, these orderers need to make consensus on the strict order of the transactions. This where the consensus algorithms may help to make orderers all agree on only one order for the transactions. As for the consensus details, please refer to the previous chapter 3.

After the consensus is reached, a unique order is there, and transactions are packed into blocks. The orderers send the blocks to all channel peers (committing peers) for final validation and commit stage. Once all transactions in the blocks are validated and correct, the blocks are committed into the peers' ledgers.

Requirements linked to the Ordering Service: DLT-SHA-16, DLT-SHA-17, DLT-SCT-02, DLT-SCT-04, DLT-SCT-55, DLT-SCT-56

6.2.7 Security Context Broker

The Security Context Broker (SCB) is responsible for offering a **trusted bridge** between the (trusted) blockchain network and the (untrusted) "*outside world*". Primarily, the SCB is responsible for **managing the process of the creation and deployment of the attestation smart contracts on the ledger**. As depicted in the overall ASSURED security process (**Error! Reference source not found.**), once the **scheduling (attestation) policies** have been compiled, as an output of the Policy Recommendation Engine, the last step of the overall security process is the deployment of this set of targeted policies over the SoS-enabled ecosystem for managing the identified risks, as well as to handle the real-time supervision and monitoring of the correct execution of these policies. This is done through the ASSURED Security Context Broker (SCB) via the use of smart contracts leveraging the designed policy-compliant blockchain infrastructure. The SCB, acting as the *trusted operator* of the produced policies, will be triggered by the Policy Recommendation Engine for converting the attestation policies into smart contract logic and further deploy the smart contract to the ledger.

More specifically, the SCB upon reception of the policies, will trigger the Smart Contract Composition Engine for creating the appropriate chain code of all the functions needed for supporting the enforcement, execution, recording and sharing of all attestation related data [2]. The composition engine will, in turn, invoke the *deployPolicy()* function for putting all the policies as part of the respective private channels. In ASSURED, we are adopting and advancing the **service-oriented** blockchain network model where functions are provided to all the devices as a service (**chaincode-as-a-service**). This is based on the use of the gRPC technology where the chaincode itself is defined and implemented in gRPC format: This enables all the devices to be able to continuously interact with the smart contract chaincode. Thus, whenever a new contract is deployed, all the devices that have been enrolled to this

Funded by the Horizon 2020 Framework Programme of the European Union

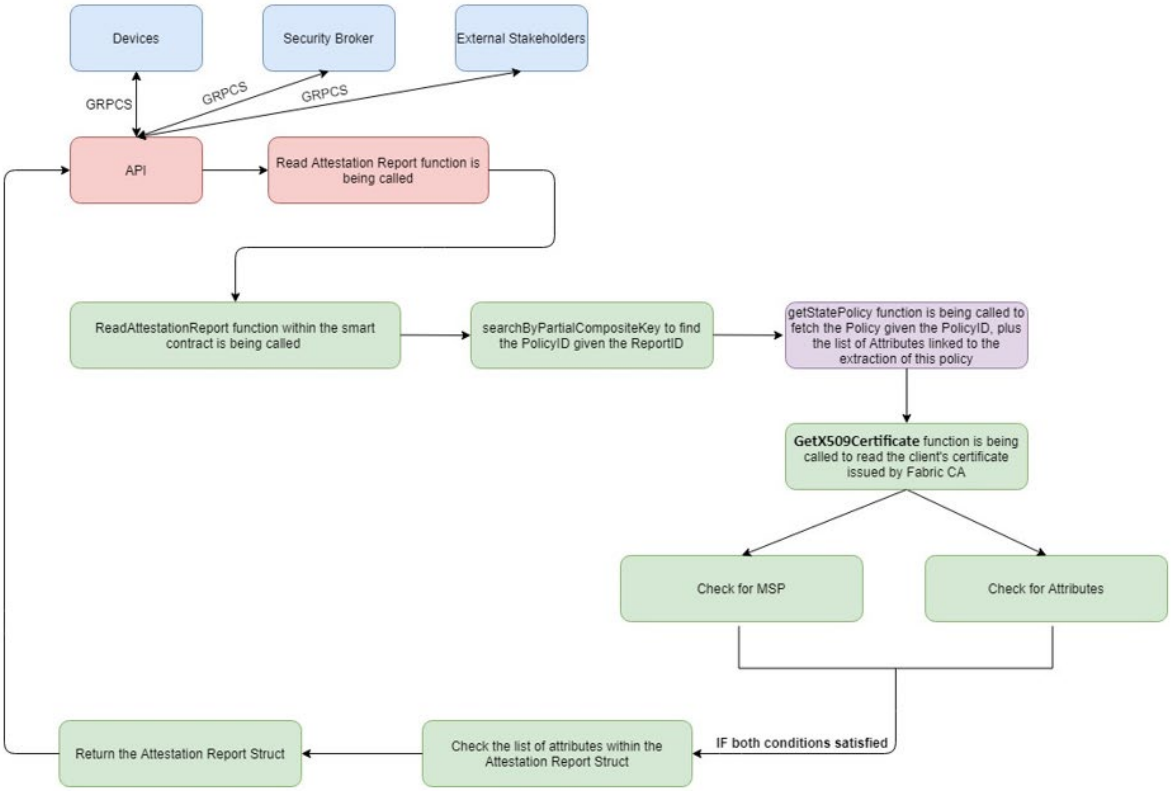


FIGURE 11: ATTRIBUTE-BASED ACCESS CONTROL PERFORMED BY THE MSP

channel will be automatically informed about this operation so they will be able to immediately check if the new deployed contract is destined for them.

After the deployment of the smart contracts, edge devices, external stakeholders and the SCB can interact with them through appropriate **functions and ledger interfaces**. In this context, however, only devices and users with the appropriate privileges will be able to access and query for specific attestation policies or results. This is enabled by the integration of advanced Attribute-based Access Control (ABAC) mechanisms supported by the SCB. Attributes are been verified during the Secure Enrolment phase (Chapter 6.4.2.1), by the Privacy Certification Authority (Privacy CA), which then credits the device a token 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 bundles. In this case, every time a device performs a query operation then this “querying function” will interact with the SCB that will invoke the GetX509Certificate function for fetching the user’s certificate from the Blockchain CA. This is then forwarded to the **Membership Service Provider (MSP)** (Chapter 6.2.8) that verifies that all required attributes are included in the user’s certificate – in which case it notifies the blockchain Peer to retrieve and send back to the user/device the request attestation data report.

Requirements linked to the Security Context Broker: DLT-SHA-11, DLT-SCT-03, DTL-SEC-08, DTL-SEC-11, DTL-SEC-12, DLT-SCT-13, DLT-SCT-14, DLT-SCT-15, DLT-SCT-16, DLT-SCT-17, DLT-SCT-21, DLT-SCT-39, DLT-SCT-46, DLT-SCT-47

Funded by the Horizon 2020 Framework Programme of the European Union

6.2.8 Membership Service Provider (ABAC service)

As aforementioned, the MSP is invoked by the SCB when there needs to be a verification of the correctness of the attributes provided by a device or an external stakeholder wishing to access some of the recorded attestation-related data.

Figure 11 depicts this Attribute-based Access Control (ABAC) offered by the MSP: To access the report one will need to request access permission via the *getAttributes()* function that performs the ABAC to check the attributes of the entities that are allowed to access the result. Based on the response of this function, there are two operations performed for the requesting entity: (i) **Get access to the attestation report stored on the ledger** - in this case, the attribute checking is done as part of the attestation report query, and (ii) **Get access to the encrypted raw traces**, linked to this attestation report, stored on the offline database. For the latter, the SCB will be able to retrieve the data and forward it to the requesting party. Recall that all system traces are encrypted by the (Verifier) device through our newly designed and decentralized ABE scheme leveraging the host TPM-based Wallet. Therefore, the data that will be received by the requesting entity can only be decrypted if its TPM-based Wallet can verify the correctness of the attributes so as to create the necessary decryption keys.

More specifically, an edge device or user, invokes the *readAttestationReport()* or *readAttestationPolicy()* function through the API Layer of the blockchain Peer. Then, the appropriate contract (on the ledger) has been identified by searching with the corresponding composite key identified (e.g., policy identifier, attestation report identifier, etc.). This also fetches the list of attributes that have been associated with this specific policy or attestation report. Then, the *GetX509Certificate()* function is been invoked for fetching the certificate – for this specific device or user – as issued by the Blockchain CA during the device/user enrolment and registration (Chapter 2.2). This list and the device's/user's certificate is forwarded to the MSP that is responsible for checking the correctness of all attributes included in the certificate. If successful, then the MSP notifies the SCB that it can proceed with giving access to the device/user to the requested data.

Requirements linked to the ABAC Service: DLT-SHA-03, DLT-SHA-04, DLT-SHA-06, DLT-SHA-07, DLT-SHA-08, DLT-SCT-15, DLT-SCT-18, DLT-SCT-19, DLT-SCT-20, DLT-SCT-21

6.3 ARCHITECTURAL BUILDING BLOCKS - DEVICE LEVEL

6.3.1 TPM-based Wallet

Blockchain wallets are used in the context of ASSURED to provide **strong device authentication**. Authentication provides assurance that protected data was not modified and that it came from an entity with access to a key value that needs to be a secret or a shared secret. Another functionality of the TPM wallet in the ASSURED framework is the **secure key storage**, each edge device is expected to have a TPM-enabled Blockchain Wallet that can securely store cryptographic key material and platform measurements that help ensure that the connected devices remain trustworthy.

In addition to this basic function of storing the keys, a trusted wallet may offer the functionality of encrypting information to ensure secure communication channels between the devices and peers. TPM wallets also control and authorise the access to private or public ledger channels based on ASSURED device authentication processes and based on the attributes of the

Funded by the Horizon 2020 Framework Programme of the European Union

device, this is done through ASSURED Attribute Based Encryption ABE mechanism that will be explained in the next chapter.

Another crucial functionality of the TPM wallet is ensuring the **integrity of Platform Configuration Registers (PCR) measurements** that represent the platform state – based on the Configuration Integrity Verification scheme presented in D3.2 [51]. As software is loaded, hashes of the binaries are extended into the PCRs. The extension corresponds to the hashing of the concatenation of the previous content of the PCR with the inputted hash, and the storing of the result in the PCR. The nature of hardware-based cryptography ensures that the information stored in the TPM is better protected than software-preserved data. The TPM Wallet will then use the TPM's PCRs to securely store the current blockchain state hash and further use the TPM's hashing accelerator to speed up hash computations as required during blockchain operations (e.g., to compute Merkle hash trees). To guarantee that only trusted and uncompromised devices can participate in the envisioned supply chain ecosystem, all involved devices will use the TPM wallet secure boot mechanism and their trust level will be continuously attested and assessed. A verifier checks if the reported TPM wallet PCR value matches with one of the accepted (White-List) PCR values then verifies the TPM's signatures on attestation data (e.g., transactions, smart contracts) that 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. Depending on the selected privacy level, a conventional or a privacy-preserving signature scheme may be employed. In the former case, a plain digital signature scheme supported by the TPM (e.g., ECDSA) will be selected, whereas in the latter case the TPM-provided Direct Anonymous Attestation (DAA) scheme can be used as a strong privacy-preserving signature scheme.

Requirements linked to the TPM Wallet: DLT-SHA-10, DLT-SCT-10, DLT-SCT-26, DLT-SCT-27, DLT-SCT-30, DLT-SCT-31, DLT-SCT-32, DLT-SCT-33, DLT-SCT-35, DLT-SCT-41, DLT-SCT-42, DLT-SCT-34, DLT-SCT-46

6.3.2 Attribute Based Encryption (ABE) Component

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 such a system, the decryption of a ciphertext is possible only if the set of attributes of the user key matches the attributes of the ciphertext. The blockchain user (being authenticated to a ledger) 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.

For instance, data encryption and access control is a topmost requirement in the "*Public Safety*" ASSURED use case and needs to be complemented by specified policies and access control mechanisms to guarantee that only valid data retrievers can decrypt them. To achieve this requirement, an assigned administrator pre-defines a policy access list that will be merged on the smart contract to auto-check if different domains'/areas' operational level's entities could have access to the operational data. Specified policies and access control mechanisms would be implemented through Attribute Based Encryption ABE schemes to guarantee that only eligible entities can retrieve data from the cloud if needed.

Funded by the Horizon 2020 Framework Programme of the European Union

For the "*Smart Manufacturing*" ASSURED use case, data is stored on IoT gateways, and the data which can be in the form of archives, snapshots, reports etc, are recorded at the respective central database. Different IoT gateways and a central database will commit data sharing behaviours. An IT administrator creates and defines an access policy control that is achieved through creating attribute tokens granted for the different entities that interns run the ABE protocol with the predefined access control trees to access the stored data based on their different levels of attributes. Any illegible entity with the required attributes can then connect to the IT infrastructure to acquire the necessary data. This would monitor the data access among the gateways and the database.

ABE is also required by the "*Smart Aerospace*" use case where the sensors within an aeroplane can share data with the Secure Server Router SSR after passing a successful on-boarding process through the ASSURED authentication mechanism. The SSR administrator needs first to define the access policy in a form of smart contract by granting access tokens that regulates the access to ASSURED private ledger. If the authentication is successful and the access policy can be satisfied, the SSR can share its data with a Ground Station Server and the latter can then share data with the analytics cloud Servers.

ABE will also be one of the main ASSURED components adopted by the "*Smart Satellites*" use case where the Ground Station GS is allowed to share threat intelligence data if it processes an access token that enables data sharing. The data access policy should be clearly defined in the smart contracts so that access policies can be checked during data sharing. ASSURED attestation mechanisms should enable CubeSat to perform attestation to the GS. The attestation results can be verified and recorded on an accessible platform for internal entities. For external entities, ASSURED DLT will enable them to read the attestation data. In this case, access control policy enforcement is required through ABE.

The TPM-based Wallet supports ABE that is mainly used to **ensure legitimate attribute-based access control to sensitive encrypted data**. A trusted authority which can be many entities as identified in the figure below (in our context it is the SCB that takes up this role), generates the user attribute keys using the authority master key. The trusted authority then sends the encrypted attribute symmetric decryption keys together with the attribute policies to the edge device which contains an embedded TPM Wallet. The TPM Wallet stores the decryption symmetric keys safely inside the TPM. **Using these keys together with the access control tree the TPM can recover the main decryption and integrity keys that will be used by the host to check the message integrity through a message Authentication Code MAC and decrypt a message respectively.** We will discuss the proposed ABE scheme as shown in Figure 12.

The Attribute Based Encryption scheme to be used in ASSURED is based on the Elliptic Curve Integrated Encryption Scheme presented in [48]. The different stages in that scheme are the following:

- **Setup**: The attribute space in the system is defined as the universe of attributes

  U = {1, 2, . . . , n}. For each attribute i ∈ U, the authority, which can be the security context broker in ASSURED framework, choose a number ti uniformly at random from Z^*_q. The public key of each attribute i is: i_i . G. The authority chooses as secret integer t uniformly at random from 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}.

Funded by the Horizon 2020 Framework Programme of the European Union

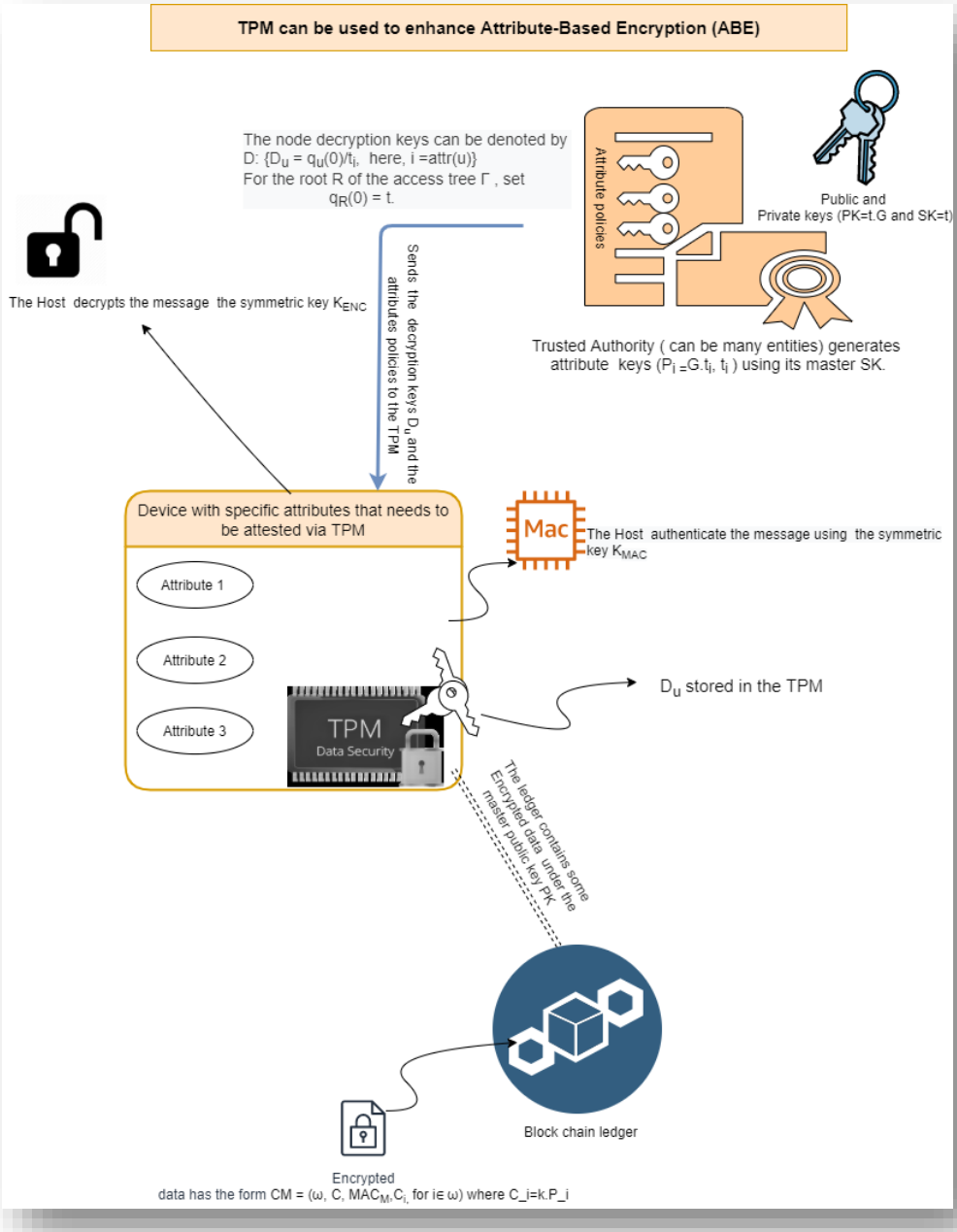


FIGURE 12: ASSURED ABE SCHEME USING THE TPM WALLET

- **KeyGeneration (Γ, MK):** A polynomial $q_u(x)$ with order of (d_u-1) should be defined by an authority for each node u in the access tree Γ in top-down manner, where d_u is the threshold of the node u. For the root R of the access tree Γ , set $q_R(0) = t$. The TPM decryption key can be denoted by D: $\{D_u = q_u(0)/t_i$, here, i = attr(u)$\}$ and t_i is the randomly

Funded by the Horizon 2020
Framework Programme of the European Union

chosen number from Z^*_q in Setup phase, t_i^{-1} is the inverse element of t_i over finite field Z^*_q .

- **Encryption:** To encrypt a message M under the set of attributes ω, an encryptor, which can be any blockchain ASSURED user such as blockchain node, randomly chooses k from Z^*_q to compute $C', C' = k \cdot PK = (K_x, K_y)$ based on the encryption scheme defined in [49].

  K_x will be the encryption key and K_y be the integrity key for a message M, C and MAC_M can be computed respectively, where C = ENC (M, K_x) is the encrypted message and MAC_M = HMAC(M, K_y) is the message authentication code.

  The cipher-text is denoted by CM = (ω, C, MAC_M, C_i, for i ∈ ω) where C_i=k.P_i.

- **Decryption (CM, D, Params):** As other ABE schemes, 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 GE or ⊥. Accordingly, for the root node R of the access tree, there should be

  $DecryptNode(CM, D, R) = q_R(0) \cdot k \cdot G = t \cdot k \cdot G = (K_x, K_y)$. If $HMAC(M, K_y) = MAC_M$, it indicates that message M is correctly decrypted and is not tampered.

Requirements linked to the ABE Component: DLT-SHA-14, DLT-SCT-01, DTL-SEC-06, DLT-SCT-22, DLT-SCT-23, DLT-SCT-24, DLT-SCT-25, DLT-SCT-26, DLT-SCT-34, DLT-SCT-35, DLT-SCT-40, DLT-SCT-41, DLT-SCT-42, DLT-SCT-43

6.4 ARCHITECTURAL BUILDING BLOCKS - ORGANISATION BOUNDARIES LEVEL

This chapter describes components that lay outside of the core DLT network (e.g., outside of the private and the public channels) but are essential to the overall operation of the DLT network and are integral parts of the overall architecture concept.

6.4.1 Smart Contract Composition Engine

The Smart Contract Composition Engine is the component that is tasked with **authoring the smart contracts (e.g., the chaincode) that will be installed and executed by the different peers, as identified in the architecture diagram** (Figure 10). This component is sitting outside of the different private and public channels of the overall ASSURED DLT network and is providing the interface to developers to design the code that will be necessary to facilitate the operation of the network.

Through this component, a developer will author the code using the appropriate language for the network (in the case of Hyperledger Fabric this will be Go or node.js) and thereafter they choose the target peers where their code shall be deployed. The overall chaincode logic is deployed in this component, taking also as input the different policies (which are then resolved

Funded by the Horizon 2020 Framework Programme of the European Union

by the SCB) and once ready the chaincode is deployed by the blockchain administrator to the different peers, e.g., the smart contracts that are to be executed in each peer are installed and made available as chaincode as described in chapter 3.

The smart contract composition engine also makes the SCB aware of the location of the different chaincode APIs, as the SCB will need to know where each API resides to call it and route the different requests to it, when it comes to performing various operations over the network which shall be executed by specific peers and specific smart contracts.

Requirements linked to the Smart Contract Composition Engine Service: DLT-SHA-05, DLT-SCT-19, DLT-SCT-20

6.4.2 Privacy Certificate Authority (Privacy CA)

As aforementioned in Chapter 2.2, the Privacy CA is responsible for verifying the attributes demonstrated by a device during its enrolment to the system network (and, thus, to the blockchain network). Attributes in this context span from the **certification that a device has a valid TPM Wallet** (by certifying the Endorsement Key (EK) for the host TPM [51]) and **validation of the correct creation of an Attestation Key (AK)**, used later on for the device's participation to the execution of respective attestation policies, to the **validation of other device attributes (e.g., correct firmware, software loaded, identifier to be part of a specific organization, etc.)**.

The singing key that is used for the TPM attestation services is referred to as Attestation Key (AK). *To preserve user privacy in some ASSURED applications ("Public Safety" use case), AK is not used as the TPM's identity*. However, any external verifier wants to make sure that given attestation information was created by a genuine TPM even if it is unidentified. Each TPM has an Endorsement Key (EK), usually certified by the TPM manufacturer, that is used to represent its identity. To create signatures, the TPM must have a valid AK credential (Cred), which is a certificate containing the public AK and a proof that the corresponding private key is bound to a genuine TPM with a valid unrevoked Endorsement Key (EK).

This proof is guaranteed by a signature on the TPM's public AK created by a trusted third party known as a "Privacy Certificate Authority (CA)". **A Privacy CA is a trusted ASSURED party that signs the public TPM Wallet's attestation key with all of its corresponding attributes**. This signature represents the TPM AK credential which will be used to prove that a TPM attestation key belongs to a valid TPM with a certified public EK. Upon creating ASSURED attestations, any external verifier will be able to verify that a TPM has a valid credential that is signed/ certified by a trusted privacy CA.

Requirements linked to the Smart Contract Composition Engine Service: DLT-SHA-09, DLT-SCT-22, DLT-SCT-28, DLT-SCT-49, DLT-SCT-53

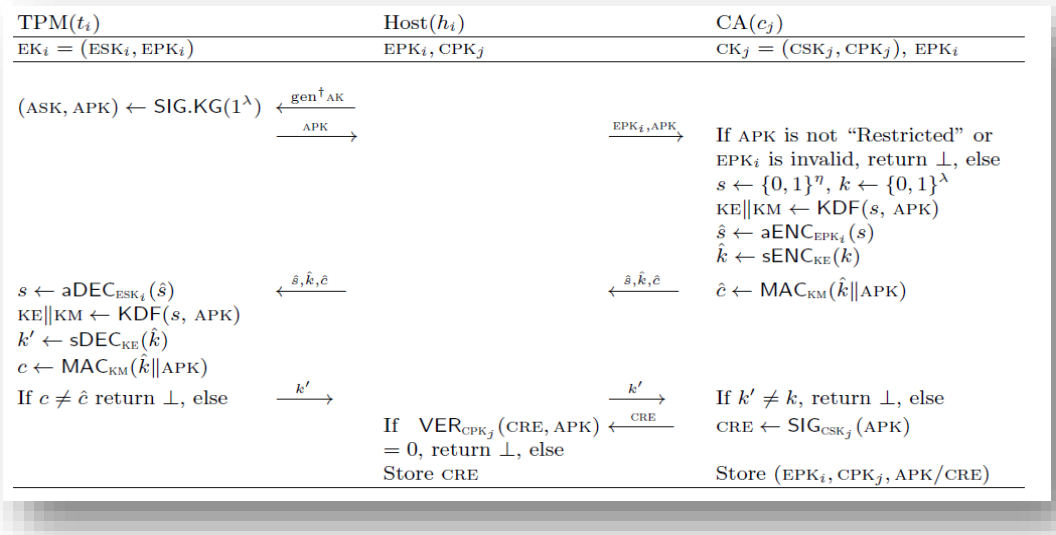
6.4.2.1 Secure Device Enrolment in ASSURED

To create signatures, the TPM must first have a valid credential (CRED), which is a certificate containing the public Attestation Key AK and a proof that AK originates from a genuine TPM. The credential is provided by a trusted party known as a "privacy CA". In the ASSURED framework, devices should make attestations through the trusted wallets with valid credentials to show that they maintain specific properties that may be required for accessing sensitive data or other applications. To meet this requirement, the Privacy CA is involved to authenticate that the AK holder is a genuine TPM to issue a credential for the AK. TPM AK credential issuing

Funded by the Horizon 2020 Framework Programme of the European Union

scheme involves three entities: a set of TPMs, a set of hosts and a Credential Provider (Privacy CA). The credential provider has a public and secret key pair (CPK; CSK), which is used for a signature scheme. Each TPM has a public and secret Endorsement Key (EK) pair (EPK; ESK), which is used for an asymmetric encryption scheme. The 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 TPM also has a public and secret AK pair, which is used for a signature scheme (either a conventional signature scheme or a DAA signature scheme).

We will explain the Attestation Certificate Authority Solution adopted by TPM 2.0 (ACAS-in-TPM-2.0) [50] as shown in Figure 13. The TPM with EK (ESK, EPK) generates a restricted Attestation Key AK (APK, ASK), which should be certified by a Certification Authority CA. The TPM then sends the public part of the attestation key APK together with the public part of the endorsement key to the CA through the host. After checking the validity of the TPM attestation and endorsement keys, the Certification authority creates a credential (signature on APK using the CA private key) using the make credential function. The CA generates a secret symmetric encryption key k that is used to encrypt the credential together with a random seed s. The CA generates a 'seed' s to a Key Derivation Function (KDF). This seed s is encrypted under the TPM endorsement encryption key EPK. The seed s is used in the Trusted Computing Group TCG-specified Key Derivation Function KDF to generate a symmetric encryption key KE and an HMAC key KM. The symmetric key KE is used to encrypt the secret key k, and the HMAC key KM provides integrity. The encrypted secret and its integrity value are sent to the TPM in a credential blob. The encrypted seed is sent as well. The encrypted secret and its integrity value are sent to the TPM in a credential blob. The encrypted seed is recovered by the TPM using ESK. The seed remains inside the TPM. The TPM unwraps the credential and returns the symmetric decryption key k to the CA through the host. This TPM operation is called "activate credential". The CA then authenticates the TPM and signs the TPM public attestation key APK, this signature corresponds to the TPM credential. The credential is then sent to the host who 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 host stores the credential and loads it whenever the platform (TPM + Host) needs to generate signatures.



Funded by the Horizon 2020 Framework Programme of the European Union

FIGURE 13: THE ATTESTATION CERTIFICATE AUTHORITY SOLUTION IN TPM 2.0 (ACAS-IN-TPM-2.0) REQUEST GENERATING A "RESTRICTED" SIGNING KEY AK

For the TPM Wallet to be engaged in ASSURED functionalities, it must have a valid credential. **The Privacy CA can verify that the TPM has a valid CA credential on the TPM public key**. If this verification is successful, then the Blockchain CA provides the necessary certificate for the TPM wallet to access the ledger as shown in Figure 14.

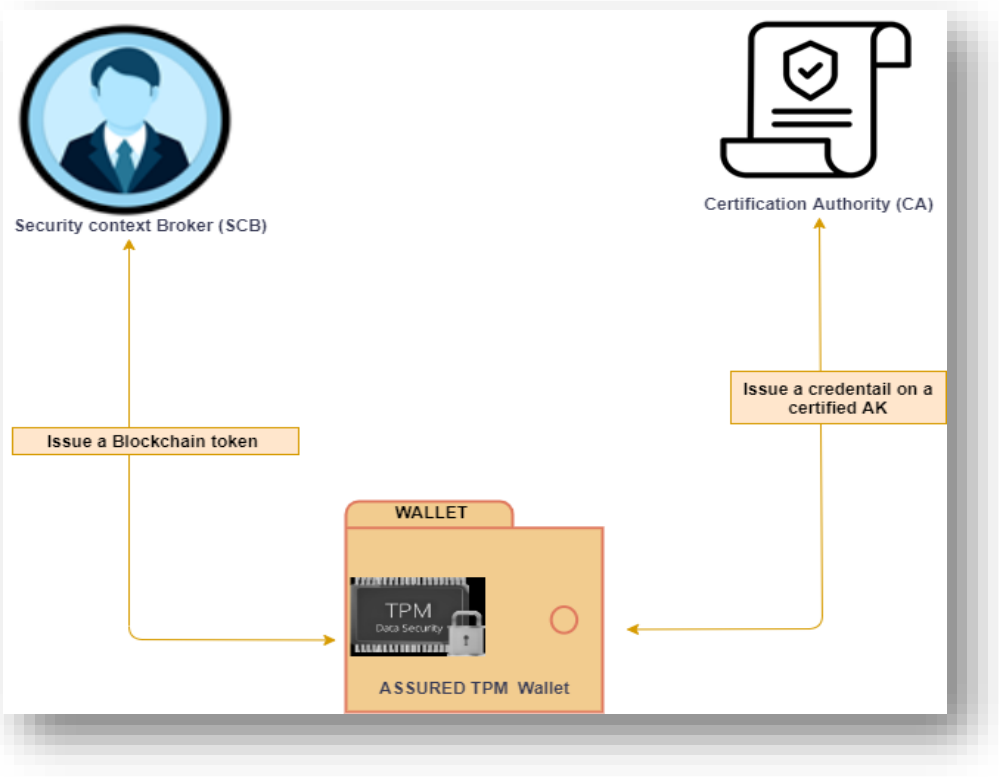


FIGURE 14: SECURE DEVICE ENROLLMENT

The protocol for obtaining a credential is intentionally *not privacy-preserving as the Blockchain CA authority needs to be aware of the wallet to be authenticated*. However, successful completion of the protocol results in the TPM wallet solely owning a valid credential that can be privacy-preserving through the execution of the Direct Anonymous Attestation (DAA) protocol [51]. In this context, depending on the privacy requirements of the device and the SoS-enabled ecosystem, the TPM Wallet, upon registration with the Privacy CA and the Blockchain CA, will create its own *pseudonyms* that are used to self-certify a TPM credential and this certification can be verified by any blockchain user. **This is called pseudonymity which is the ability of a TPM wallet to use a resource or service without disclosing the user's identity while still being accountable for that action**, i.e., any Revocation Authority (RA) can then remove misbehaving wallets without revealing the TPM wallet's identity. The concept of pseudonyms in the context in ASSURED is explained in deliverable D3.2 [51].

6.4.2.2 Certified Restricted Key Attributes

Each key has attributes, which are set at creation of the key, include the following: Use attribute such as signing or encryption key and type attribute that identifies if the key is a symmetric or

Funded by the Horizon 2020 Framework Programme of the European Union

asymmetric key. There are also some attributes that restrict the TPM key functionalities. For example, the TPM can only use its signing key to generate signatures if some conditions are satisfied. For instance, the TPM signature over a digest restriction is essential when creating PCR quotes for ASSURED attestation services. For instance, a user could generate a digest of any PCR values and use a non-restricted key to sign it. The user could then claim that the signature was a quote. However, the relying party would observe that the key was not restricted and thus not trust the claim. A restricted key provides assurance that the signature was over a TPM generated digest. Trusted Platform Modules protect the use of keys through two distinct methods: password-based and policy-based. In password-based protection, a key cannot be used unless the correct password is provided. In policy-based protection, a policy or a set of policies must be defined upon the creation of the TPM signing key and should be later satisfied whenever the key is used. The privacy CA authority that certifies the TPM attestation/signing key will clearly certify the corresponding key restricted attributes when generating the key credential. Such certified key attributes are crucial in the TPM key management as they can prevent many attacks by enforcing restrictions on the TPM key functionalities.

For example, in ASSURED we have a particular authentication challenge. We need to send Traces from the TCB (secure world) through the TSS (insecure world) to the TPM (secure world) for signing. We must guarantee that our signing key cannot be misused in any way by the insecure world. We protect this by using a policy which states that the TPM signing key can only be used if the Tracer authorizes it. This can be achieved through a TPM policy called PolicySigned. PolicySigned states that a valid tracer's signature on the traces must be provided using an authorizing key whenever the TPM wants to create a signature on the traces provided by that tracer. PolicySigned restricted attribute is certified by the privacy CA upon the creation of the credential on the TPM signing key.

6.4.3 Blockchain Certificate Authority

Before proceeding to the Blockchain CA, we here review the digital certificates used in the blockchain networks. A digital certificate is a digital form of document that contains a set of attributes for the document holder, so as to convert the holder's ID details into the certificate. This important and identity related certificate should be kept securely in cryptographic way, which is usually via digital signature. Specifically, to make sure that the certificate has not been tempered, a digital signature should be signed on the certificate, so that anyone else can use the corresponding public key to verify if the signature is still valid. If so, it means that the certificate is validated and trustworthy.

In this context, the issuer of the certificate is known as certificate authority. This role will be set up in the ASSURED blockchain network, so-called Blockchain CA. This CA will issue the digital certificates for all the entities within the network (i.e. digital identity form with the CA's signature), e.g., Orderer, Devices, and anyone can verify its validity via the CA's public key.

This general CA must be set in the system deployment stage and should be fully trusted. In ASSURED, we will have this CA as root CA, and we may consider using sub CAs for different DLT layers, for example, a channel could have its own channel CA. And the root CA and the sub CAs will form chains of trust, meaning that a digital certificate of a device/peer/orderer within a given channel is signed by the channel sub CA and the certificate of the sub CA is signed by the root CA. A new device is able to be issued a digital certificate by the channel sub CA, and with this certificate, its attributes can be validated while it is joining the channel.

Funded by the Horizon 2020 Framework Programme of the European Union

The ASSURED Blockchain CAs will manage a special list - called certificate revocation list, containing a list of references to certificates that are revoked. This list can help certificate verification on access control mechanisms, e.g., MSP and ABAC, check if a given certificate is compromised easily. Note that if the revocation list is not checked, then a verifier may suffer from the risk that the certificate may be misused or stolen by some impersonator. This is so because the verifier can use the corresponding public key to successfully validate the signature on the certificate; from this, it cannot tell if the certificate is revoked.

Requirements linked to the Blockchain CA: DLT-SHA-10, DLT-SCT-10, DLT-SCT-18, DLT-SCT-26, DLT-SCT-29, DLT-SCT-48

6.4.4 Off Chain Storage

The off-chain storage is essentially the storage facility that holds different files originating from attestation operations, and other information that is not to be stored on the ledgers. The ledger will hold pointers to the specific location of the Off Chain Storage where data relevant to a transaction is residing, and those pointers will be either plaintext or encrypted, based on the weather these are stored in the private ledger (plaintext pointers) or in the public ledger (encrypted pointers). As such the link between the information on the ledgers and on the Off Chain Storage will be done by the SCB interpreting those pointers and being the one responsible to store and fetch the data from the Off Chain Storage facility. As such, the Off Chain storage will not be directly exposed to any entity for security purposes, and all interactions with it will be done via the Security Context Broker.

This facility will be based on a noSQL database (document database) such as MongoDB[46] and will include the encrypted data that will be coming from the different devices (through the Security Context Broker). These data will be encrypted using the ABE Scheme based on the keys to be stored in the TPMs of each device, and a pointer of the storage location of each such entity will be stored on the ledger.

Requirements linked to the Off Chain Storage: DLT-SHA-15, DLT-SCT-03, DLT-SCT-05, DTL-SEC-08, DLT-SCT-13, DLT-SCT-37

6.4.5 Indexing Service

To allow the performant querying of the off-chain storage by the Security Context Broker, a mainstream indexing service will be deployed, to allow fast and responsive queries.

This is required as it is expected that the number of devices that might become members in an ASSURED DLT network, and the attestation outputs they generate will continuously increase, and thus such a service will provide the necessary performance guarantees to make the overall system responsive and performant.

Requirements linked to the Indexing Service: DLT-SHA-15, DLT-SHA-17, DLT-SCT-05, DLT-SCT-13

[46] https://www.mongodb.com/

Funded by the Horizon 2020 Framework Programme of the European Union

6.5 MAPPING ARCHITECTURE BLOCKS TO REQUIREMENTS

This chapter provides a summary table linking the different components of the ASSURED Architecture with the requirements identified in chapter 3.

Funded by the Horizon 2020
Framework Programme of the European Union

TABLE 13: ARCHITECTURE BLOCKS MAPPED TO REQUIREMENTS

Component	Relevant Data Sharing Requirements	Relevant Security, Trust and Privacy Requirements
DLT Peers	DLT-SHA-01	DLT-SCT-25, DLT-SCT-257, DLT-SCT-50, DLT-SCT-53, DLT-SCT-55
Chaincode Component	DLT-SHA-05, DLT-SHA-16, DLT-SHA-17	DLT-SCT-04, DLT-SCT-09, DLT-SCT-16, DLT-SCT-19, DLT-SCT-20, DLT-SCT-51, DLT-SCT-52, DLT-SCT-54
Private and Public Ledger	DLT-SHA-01, DLT-SHA-02, DLT-SHA-03, DLT-SHA-04, DLT-SHA-06, DLT-SHA-07, DLT-SHA-08, DLT-SHA-16, DLT-SHA-17	DTL-SEC-06, DTL-SEC-07, DLT-SCT-13, DLT-SCT-50, DLT-SCT-51, DLT-SCT-52, DLT-SCT-54
Searchable Encryption Component	DLT-SHA-18	DLT-SCT-01, DTL-SEC-06, DLT-SCT-14, DLT-SCT-17, DLT-SCT-36, DLT-SCT-38, DLT-SCT-39, DLT-SCT-45, DLT-SCT-47
API Layer	DLT-SHA-12, DLT-SHA-13	DLT-SCT-04
Ordering Service	DLT-SHA-16, DLT-SHA-17	DLT-SCT-02, DLT-SCT-04, DLT-SCT-55, DLT-SCT-56
Security Context Broker	DLT-SHA-11	DLT-SCT-03, DTL-SEC-08, DTL-SEC-11, DTL-SEC-12, DLT-SCT-13, DLT-SCT-14, DLT-SCT-15, DLT-SCT-16, DLT-SCT-17, DLT-SCT-21, DLT-SCT-39, DLT-SCT-46, DLT-SCT-47
Membership Service Provider (ABAC service)	DLT-SHA-03, DLT-SHA-04, DLT-SHA-06, DLT-SHA-07, DLT-SHA-08	DLT-SCT-15, DLT-SCT-18, DLT-SCT-19, DLT-SCT-20, DLT-SCT-21
TPM Wallet	DLT-SHA-10	DLT-SCT-10, DLT-SCT-26, DLT-SCT-27, DLT-SCT-30, DLT-SCT-31, DLT-SCT-32, DLT-SCT-33, DLT-SCT-35, DLT-SCT-41, DLT-SCT-42, DLT-SCT-34, DLT-SCT-46
Attribute Based Encryption (ABE) Component	DLT-SHA-14	DLT-SCT-01, DTL-SEC-06, DLT-SCT-22, DLT-SCT-23, DLT-SCT-24, DLT-SCT-25, DLT-SCT-26, DLT-SCT-34, DLT-SCT-35, DLT-SCT-40, DLT-SCT-41, DLT-SCT-42, DLT-SCT-43
Smart Contract Composition Engine	DLT-SHA-05	DLT-SCT-19, DLT-SCT-20
Privacy CA	DLT-SHA-09	DLT-SCT-22, DLT-SCT-28, DLT-SCT-49, DLT-SCT-53
Blockchain CA	DLT-SHA-10	DLT-SCT-10, DLT-SCT-18, DLT-SCT-26, DLT-SCT-29, DLT-SCT-48
Off Chain Storage	DLT-SHA-15	DLT-SCT-03, DLT-SCT-05, DTL-SEC-08, DLT-SCT-13, DLT-SCT-37
Indexing Service	DLT-SHA-15, DLT-SHA-17	DLT-SCT-05, DLT-SCT-13

Funded by the Horizon 2020 Framework Programme of the European Union

7 ASSURED DLT OPERATIONS EXPLAINED

This chapter discusses the core operations that are to be performed over the ASSURED DLT network, based on the designed architecture and the security, privacy, and trust requirements of the project and of the different demonstrator cases.

In this context, we distinguish between the following core operations:

- Deploying a smart contract over the DLT network and transforming it into the chaincode that is to be executed in the different peers of the SoS-enabled ecosystem;
- Managing the membership of entities in the private/public channels and granting the access for performing different transactions;
- Placing the results of attestation processes in the private and public ledgers (respective metadata) for making them discoverable from other entities that possess the required keys to find them and retrieve them;
- Querying the ledgers for discovering attestation results that are stored in the DLT network, using searchable encryption methods;
- Extracting attestation results from the DLT network infrastructure.

The following subchapters provide the sequence diagrams for the above-mentioned operations, as well as narrative descriptions of the flow in each case. Specific implementation aspects relevant to each case will be investigated in the frame of the following tasks of WP4 and of other development activities of the ASSURED project.

7.1 SMART CONTRACT DEPLOYMENT

Once the scheduling (attestation) policies have been compiled, the next step in the security process is the **deployment of this set of targeted policies to the respective edge devices** for managing the identified risks, as well as to handle the real-time supervision and monitoring of the correct execution of these policies. This is done through the **ASSURED Security Context Broker (SCB)** via the use of **smart contracts** leveraging the designed policy compliant blockchain infrastructure. The SCB, acting as the *trusted operator* of the produced policies, will be triggered by the Policy Recommendation Engine for converting the attestation policies into smart contract logic and further deploy the smart contract to the ledger.

More specifically, as can be seen in Figure 15 and was described in Chapter 5.2.2, the SCB upon reception of the scheduling attestation policies it requests from the Smart Contract Composition Engine to convert them to chain code logic based on the functions defined in D2.2 [2]. The refinement is based on programming scripts using the Solidity language. The script will be further converted into a chaincode which can be read and understood by all blockchain Peers. Once this is done, the Smart Contract Composition Engine notifies the Ordering Peer to proceed and deploy this attestation smart contract as a transaction on the ledger. Recall that since in ASSURED we are adopting the **chain-as-a-code paradigm**, this means that upon deployment all edge devices (registered to this specific private channel) will be notified of this operation so that they can check whether the newly deployed policy is destined for them. We note that this event creation can be monitored by the SCB, just in case the peers fail or ignore to do so.

Funded by the Horizon 2020 Framework Programme of the European Union

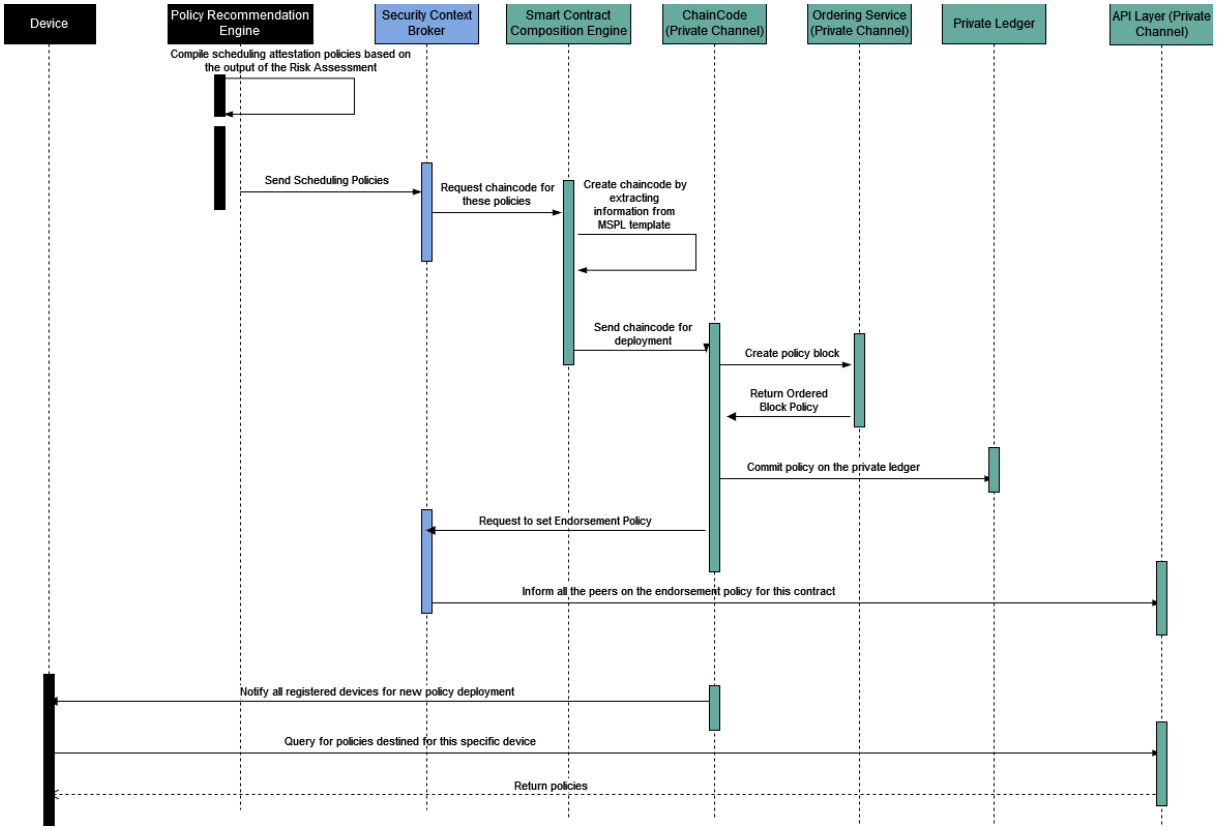


FIGURE 15: SMART CONTRACT DEPLOYMENT

One important additional step that was performed, by the SCB, prior to the deployment of the attestation smart contract is the definition of the appropriate **endorsement policy**. This essentially dictates which of the peers (as part of the same private channel) will need to validate the correct execution of an attestation contract (e.g., verification of correct signing by the Prover/Verifier, correct checking against appropriate trusted reference values, etc. by executing the *attestationVerification()* function [2]) prior to the result been recorded on the ledger. By default, all the peers are usually accounted for been able to perform such an extract validation. However, in ASSURED, in some scenarios, we will define the need of a peer equipped with a trusted component – such as TPM – for been able to correctly verify the (DAA) credentials used by an edge device when signing their traces and attestation results.

7.2 JOINING/ACCESSING A PRIVATE/PUBLIC CHANNEL

In the ASSURED blockchain framework, we have a private and a public channel (for an organisation) to maintain access control over internal and external parties. In a public channel, there is a public ledger that stores (encrypted) meta data (to the attestation report) and the (encrypted) pointers; while in a private channel, the private ledger maintains hash values of attestation report, unencrypted pointers and other related data. And the external parties, like data users outside the organisation, can first scan the metadata (via the use of searchable encryption) on the public ledger, and if they find something interesting, they will need to turn to the security context broker (SCB) for decrypting the (encrypted) pointers and further ruse the pointers to read the detailed data stored on the cloud-based backend. An internal device within the organisation is able to join a private channel using its credential, so as to read and

Funded by the Horizon 2020 Framework Programme of the European Union

write data on the private ledger. To achieve the above, both entities have to enable themselves to pass attribute and policy check-up via the smart contract by the SCB and the ABAC service. We note that ABAC in the ASSURED blockchain framework is a service supported by both the SCB and the blockchain peers.

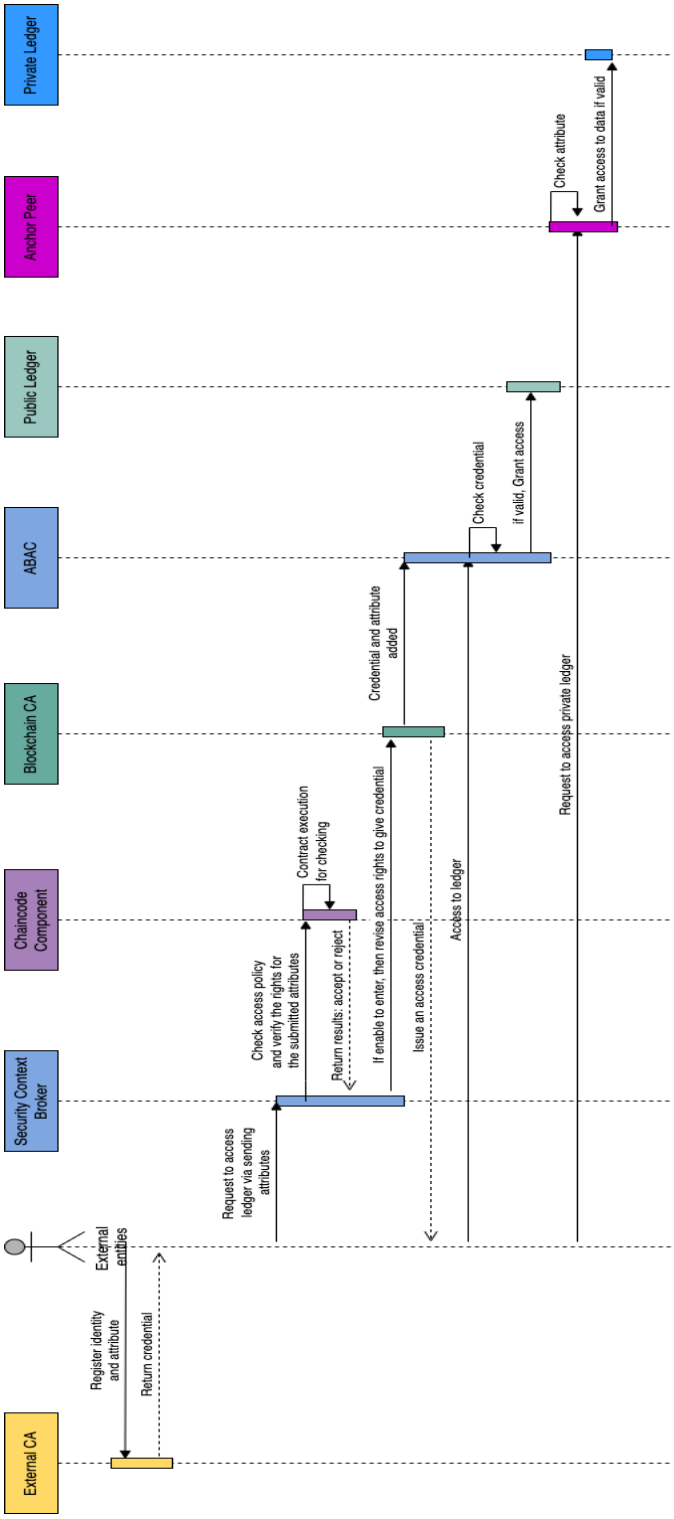


FIGURE 16: PUBLIC CHANNELS (LEDGER) ACCESS

Funded by the Horizon 2020 Framework Programme of the European Union

7.2.1 Providing Access to External Entities to the Public Channel

This type of users will first obtain **certification from an external CA (Privacy CA)** (Figure 16). Then the users will be able to send a request to the SCB with its attributes, so that the SCB will trigger the execution of smart contract appropriate function to verify if the users' attributes match the predefined access policy (through the MSP as described in Chapter 6.2.8). If they fit in the policy, the SCB will request the Blockchain CA to grant the users the corresponding credentials. In this case, the **ABAC is one of the functionalities and services supported by the SCB and the public ledger peers**. Later, when the users with the granted credentials would like to get into the channels, they are authenticated directly to the peers. And if the credentials are valid, the ABAC service enables the users to access the public channel and the corresponding ledger of a target organisation. After entering the channel, the users are allowed to use the **searchable encryption component (via the SCB) to search over their preferred keywords**. If the users locate some useful information related to the (encrypted) metadata stored on the public ledger (recall that in which metadata could be encrypted and the users locate them via searchable encryption component), they will send pointer access request to the SCB so that the SCB will decrypt the pointers on the public ledger for the users. Note that here the SCB will be able to record the data sharing statutes on the public ledger as well. And those pointers can help external users to gain access to the data (note which is a pointer with a storage address back to the cloud-based backend) stored on the cloud-based backend. In our ASSURED context, we further design a role called - the anchor peer - who enables the users to access the private ledger, say for private ledger data auditing, as long as the users are valid, via attributes checking (from the chaincode component - by the SCB, i.e., policy-based smart contract).

7.2.2 Providing Access to the Private Channel to (new) devices within an Organisation

To firstly join the private channel belonging to the organisation (Figure 17), a new device (hosting a TPM) will first send attributes to the Privacy CA so that the TPM can verify its attributes by executing the secure enrolment phase described in Chapter 6.4.2.1. Then it will send a request to the Blockchain CA for registration so that it can get the correct certification, and the TPM Wallet will provide the verification token (issued by the Privacy CA) to the CA. If the Privacy CA authentication is valid, the Blockchain CA will issue a new credential (public/private key pair) to the device and its host TPM Wallet will store the credential securely. Later, the ABAC service supported by the peers will check the device's credential provided by the TPM Wallet when the device is going to access the private ledger. If the credential is valid, the device can join the private channel and access to the private ledger.

7.3 PLACING ATTESTATION RESULTS TO THE LEDGERS

Placing the result of an attestation on the ledger is one of the core operations of the DLT network within ASSURED, and as Figure 18 shows is a rather complex process. The whole process is initiated once an attestation is executed between devices and there is the need for the verifier to store the results of this operation to the ledger, whether these concern a successful operation, or a failed operation. It is in the latter case where more data is stored on the off-chain storage, (e.g., the attestation log files), as in the case of a successful operation only an acknowledgement message of the type "Successful Attestation" shall be recorded.

Funded by the Horizon 2020 Framework Programme of the European Union

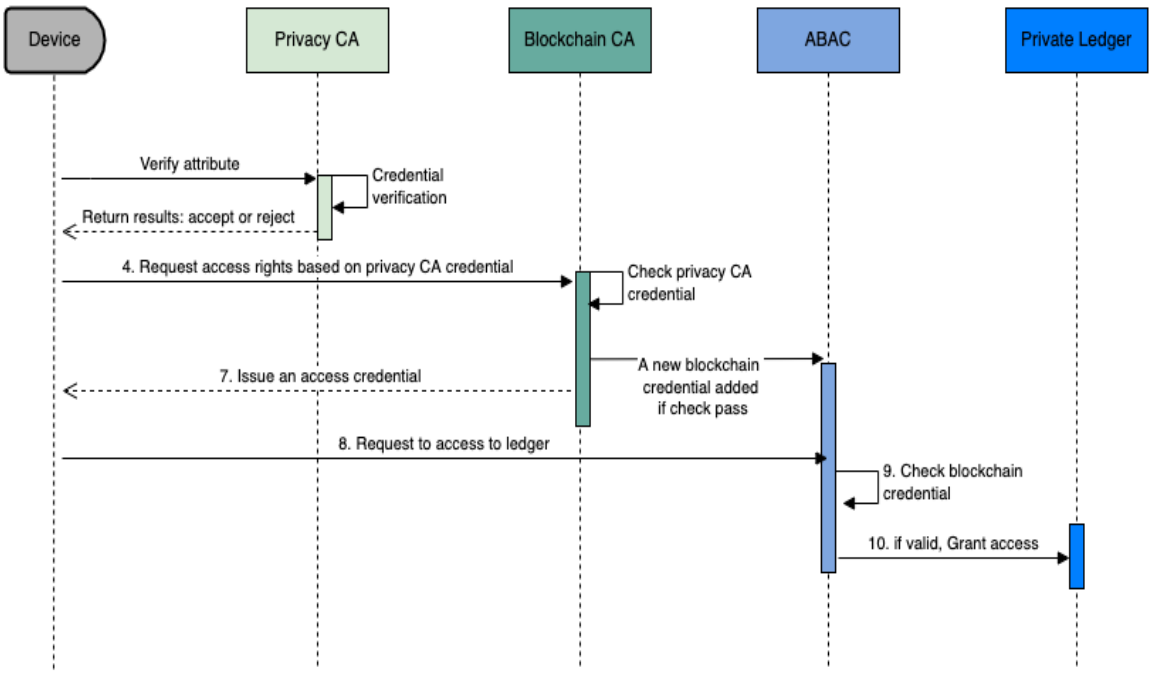


FIGURE 17: PRIVATE CHANNELS (LEDGER) ACCESS

As such, following the execution of an attestation, the device prior to sending over the different information to the DLT network is encrypting the attestation output data using its TPM Wallet ABE key, to transform the results into an encrypted blob that can be only read by entities that possess the appropriate attributes. In this manner, even if an entity gains access to the ledgers and to the off-chain storage facility, not having the appropriate attributes will prevent it from decrypting the ciphertext, and those deeming the attestation result unusable. In case the attestation has been successful, no encryption is taking place, as there is no data to encrypt.

Following the encryption step, the device is asking access to the private channel where it belongs, being verified by the ABAC service, that resolves whether the device is part of the network and of that specific private channel and returns an access permission that is then used by the device for entering the network. It is noted that this operation (and all the ones which are discussed in the other workflows that have to do with access control) is transparent to the devices and the overall access permissions that are handled by ABAC are part of the overall Security Context Broker.

The next step is for the device to send over the information it has generated to the private channel. This is performed by the device communication with the API Layer (residing within a Peer in the private channel). The core information sent by the devices is the attestation data (which will be forwarded to the private ledger), as well as the encrypted attestation log files, in case the attestation process has failed, otherwise a simple “Successful Attestation” string is sent, alongside with other information like the Device ID, a serial number for each attestation process, the timeslot of the attestation execution, etc, so that a full view of the overall attestation process is reflected in the ledger. At that point, it is the API layer that is transforming the input coming from the device to output that can be sent to the chaincode and to the Security Context Broker. Though the device could directly interact with the Security Context Broker, this is not favourable as this would entail further access control requests while there would be the need to set up an asynchronous broker for notifying the device when to send data over to the

Funded by the Horizon 2020 Framework Programme of the European Union

Security Context Broker (for performing in turn transactions over the public channel) once the private ledger operations have finished).

The next step is storing the hash of the attestation in the private ledger. As such, the chaincode component receives the hash from the API layer, executes the transaction and sends it over to the Ordering Services. The latter is returning the ordered transaction block back to the committing peers, which do commit the block in the private ledger, and the block's address is returned to the API layer, signalling that the transaction has been successfully executed in the private channel and that the flow can continue with the operations that need to be performed over the public ledger.

Funded by the Horizon 2020
Framework Programme of the European Union

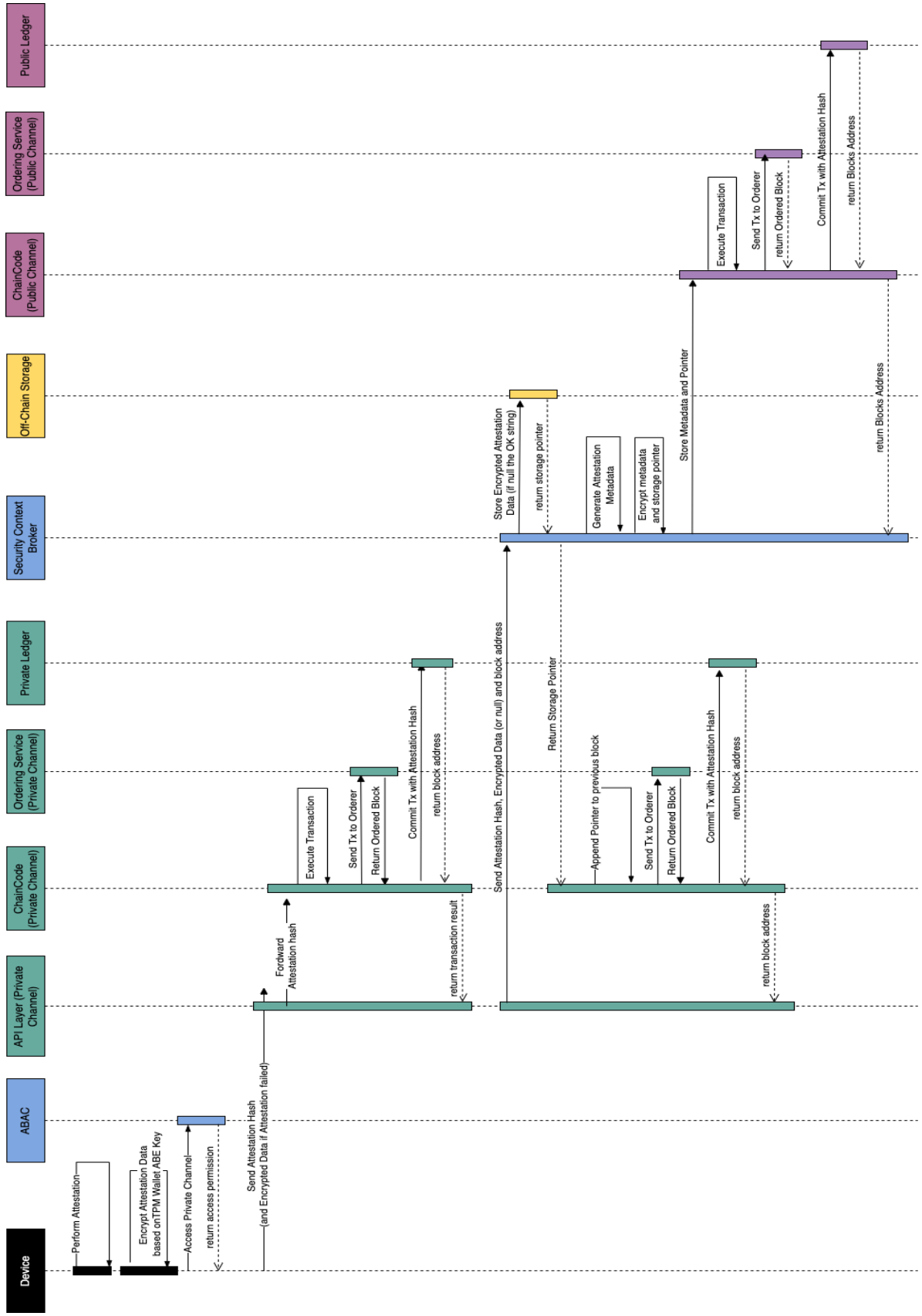


FIGURE 18: SEQUENCE DIAGRAM - PLACING ATTESTATION RESULTS TO THE LEDGERS

Funded by the Horizon 2020 Framework Programme of the European Union

Following the successful recording of the transaction in the private ledger, the API layer forwards the hash, the block's address and the encrypted ciphertext of the attestation files (or the "Successful Attestation" string) to the Security Context Broker, as there is the need to palace these information in the public ledger, and then return back the pointer to the stored attestation data to be stored in the private ledger, to be directly discoverable by the entities that belong in the same channel, without the need to go through the Searchable Encryption component of the public channel. As such, the Security Context Broker is storing the encrypted ciphertext in the Off-Chain Storage facility and gets back a pointer to the exact storage location of the data.

The returned pointer is sent back to the API Layer of the private channel, to perform another transaction in the ledger where this pointer will be appended to the previous transaction, following the exact same flow of executing and storing a transaction in the private ledger.

At the same time, the Security Context Broker will initiate the operations of storing this pointer also in the public ledger. However, as the information to be stored in the public ledger should be encrypted (and will be queried through the Searchable Encryption component), the Security Context Broker should produce the required information that will be used for facilitating the latter component. As such, metadata relevant to the transactions to be stored are generated by the Security Context Broker, which are then, together with the pointer to the off-chain storage facility, encrypted and transformed into a searchable index to be used by the Searchable Encryption component.

The operation of storing this information on the public ledger is the exact analogous of that in the private channel, where the Security Context Broker will invoke a chaincode that will be tasked to execute a storing transaction, forward it to the ordering services which will in turn return an ordered block that will be then stored by the peers in the public ledger.

7.4 QUERYING THE LEDGER FOR ATTESTATION RESULTS

Querying the ASSURED ledger for retrieving attestation results can be performed by devices belonging to the private channels (or that are permitted to access these channels, for example an auditing organisation that would like to verify specific attestation evidence), as well as devices that belong to the public channel (either within the organisation but not members of the same private channel, or them being external entities).

7.4.1 Private Ledger Querying

The first case, which we refer to as "Private Ledger Querying", concerns the case of the requestor belonging to the same private channel in which attestation data has originated from. The overall sequence of flows is shown in Figure 19.

Funded by the Horizon 2020
Framework Programme of the European Union

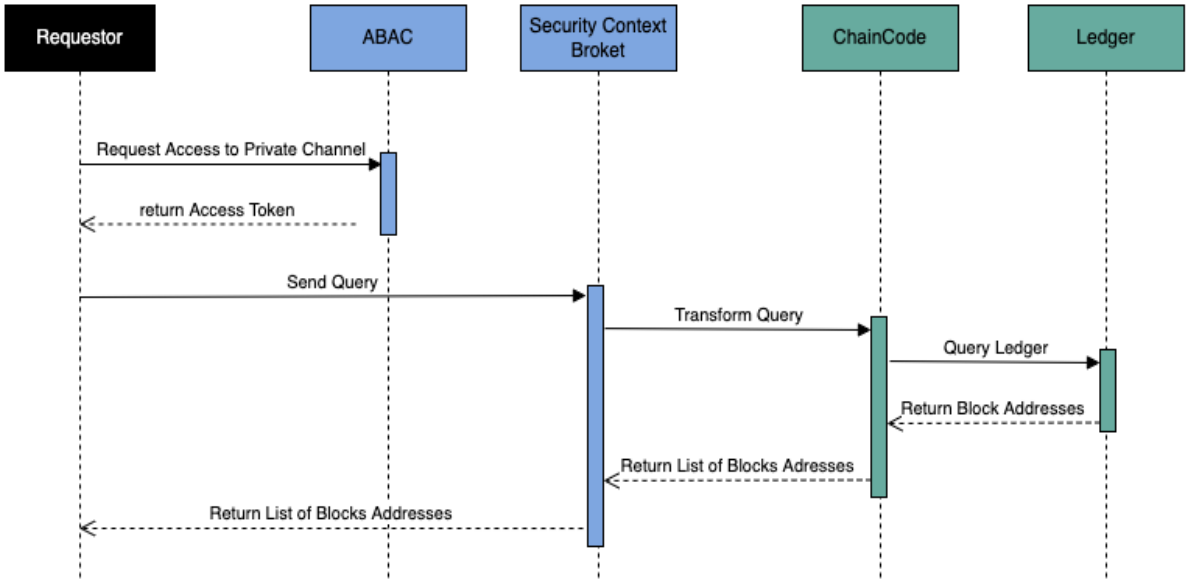


FIGURE 19: SEQUENCE DIAGRAM - PRIVATE LEDGER QUERYING

A device that wishes to discover attestation results first needs to get access on the DLT network. Therefore, it sends an access request to the ABAC and retrieves an access permission. Once it is allowed to access the network, a search query is sent to the SCB that is handling the requests of the private channel. The SCB can resolve whether the requestor belongs to the same private channels from where the attestation result was generated from. If this is not the case, then the query is forwarded to the SE Component of the Public Channel (see following case where we discuss the Public Ledger Querying methods).

In case the requestor is a member of the same private channel, based on the resolution of the SCB, then the latter is forwarding the search query to the chaincode to perform the search over the ledger's data. The results of the ledger's query are then retrieved, revealing to the requestor the address of the block where the information he seeks lies in.

7.4.2 Public Ledger Querying

The second method of querying supported, which we name as "Public Ledger Querying" has to do with searching information over the public ledger (as a result of either not finding the information requested in the private ledger (in case of inter-organisational entities), or of querying for the attestation results of devices belonging to another organisation). The sequence of flows in this case is shown in Figure 20.

Funded by the Horizon 2020 Framework Programme of the European Union

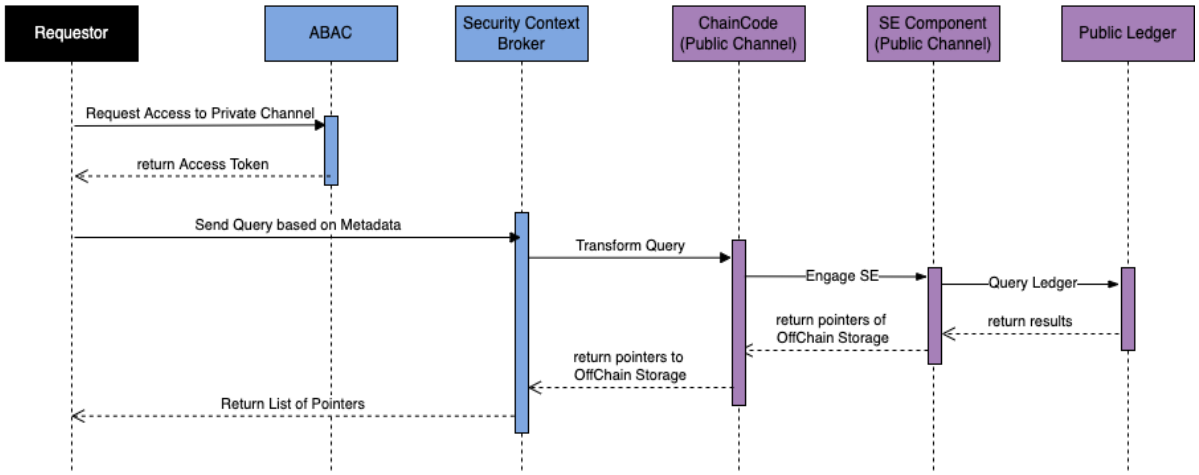


FIGURE 20: SEQUENCE DIAGRAM - PUBLIC LEDGER QUERYING

In this case, an external entity (or an entity not belonging to the same private channel where the attestation was produced) asks the ABAC to access the DLT Network. The ABAC after the necessary checks allows the entity to access, and the latter is sending over its search query to the SCB. The SCB transforms the query into chaincode input that is forwarded to the Peer running the public channel. In turn, the chaincode is engaging the Searchable Encryption Component (e.g., a subset of chaincode methods authored specifically for performing searchable encryption), that is executing the query over the ledger.

Searching over the encrypted metadata results in the generation of a list of results that are returned to the SCB. The SCB transforms the pointers to the off-chain storage locations into strings that cannot be guessed by the requestor prior to sending them back to them. In this way, it is only the SCB that can map such a string to an actual storage location in the off-chain storage, and this method aims to discourage requesters from guessing storage pointers and perform brute-force attacks for collecting attestation data (even if they are encrypted).

7.5 EXTRACTING ATTESTATION RESULTS FROM THE LEDGERS

Downloading of attestation data is a process that is always involving the SCB as FIGURE 21 depicts. We distinguish two different cases as shown below, namely extracting information from either the private ledger, or from the off-chain storage facility.

7.5.1 Extracting Attestation Evidence from the Private Ledger

In the case of extracting information from the private ledger, an entity (being it a device or another entity that is permitted to access the private channel), first requests to access the channel by communicating with the ABAC service. Once permission is granted, then a download request including the address of the block under request is sent to the SCB, who sends this request to the chaincode and the ledger is queried, returning the contents of that specific block. Those are then forwarded by the SCB back to the requestor.

It is noted that in the case of Private Ledger Querying, the pointer to the off-chain storage facility is not encrypted as there is no need to put burden on the Searchable Encryption Component to decrypt pointers that are to be used internally in the organisation. Moreover, in the unlikely case a device gets in a non-permitted way access to the private ledger bypassing

Funded by the Horizon 2020 Framework Programme of the European Union

the ABAC layer, knowing the pointer would allow it to only download the attestation files, but cannot decrypt them unless its TPM wallet has the appropriate attribute keys.

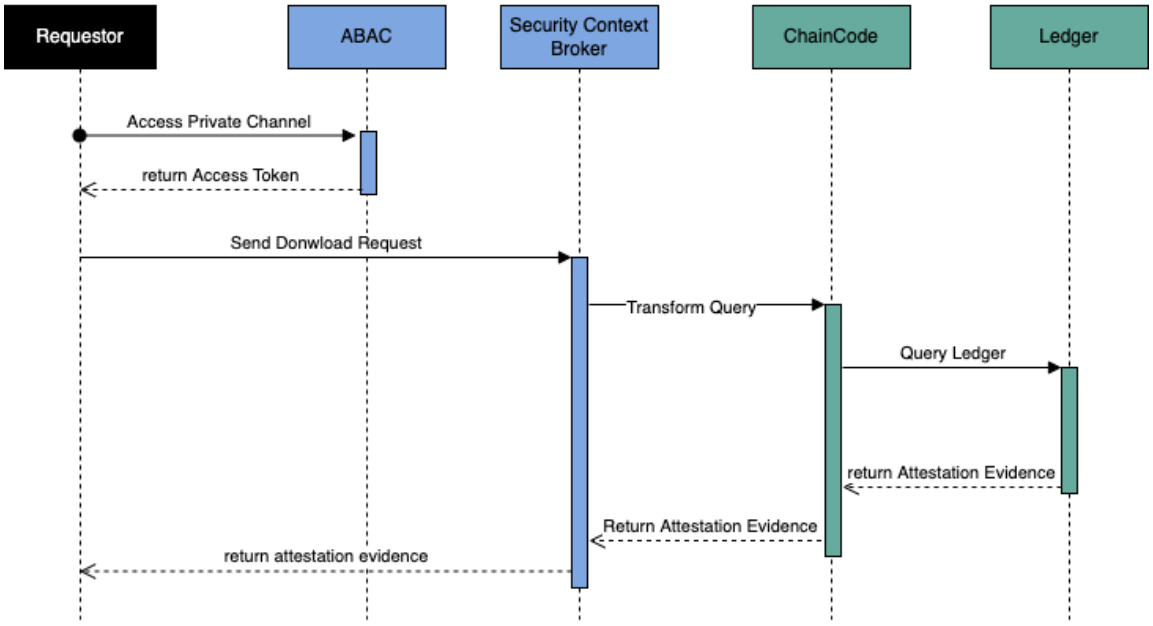


FIGURE 21: SEQUENCE DIAGRAM - EXTRACTING ATTESTATION EVIDENCE FROM THE PRIVATE LEDGER

7.5.2 Extracting Attestation Data from the Off-Chain Storage

The task of extracting encrypted information from the Off Chain Storage facility can be initiated by any stakeholder that is part of the public network of ASSURED (Figure 22). To do so, the entity needs to first get access to the DLT network to communicate with the SCB. For doing that, the entity requests access from the ABAC and once access is granted, it sends a download request to the SCB, using a value it knows as being the storage pointer. The SCB resolves this value to the actual pointer and fetches the file from the Off-Chain Storage. Finally, the file is sent back to the requestor by the SCB.

Once the file is delivered to the Requesting Entity, it can be decrypted by the ABE engine supported by the TPM, using the keys stored in the TPM wallet of the entity, following the decentralised principle of ABE used in ASSURED.

Funded by the Horizon 2020 Framework Programme of the European Union

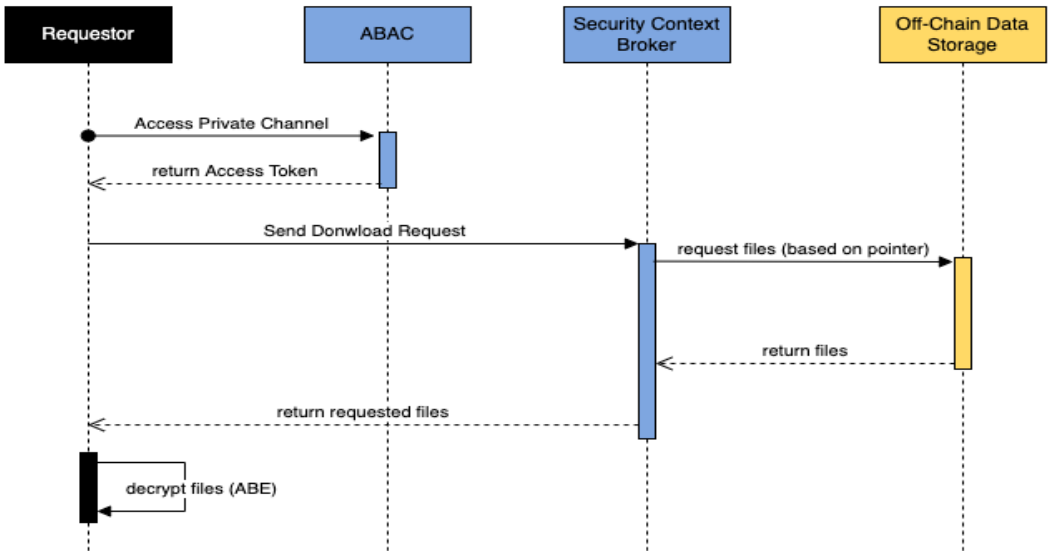


FIGURE 22: SEQUENCE DIAGRAM - EXTRACTING ATTESTATION RESULTS FROM THE OFF-CHAIN STORAGE

Funded by the Horizon 2020
Framework Programme of the European Union

8 CONCLUSIONS

In principle, as identified in this deliverable the ASSURED DLT infrastructure plays a vital role for the overall system, as it servers most of the data sharing as well as security, trust and privacy guarantees that would be offered by ASSURED to the engaged stakeholders.

As identified in this deliverable, the work for setting up the ASSURED DLT will be based on the Hyperledger Fabric technology and based on the long list of requirements identified the implementation tasks of the project will work towards delivering the various components and mechanisms/algorithms that are listed in this document. Out of those, the core innovations to be provided as part of the ASSURED DLT network are the following:

- a novel method to utilise DLTs to enable distributed and collaborative data sharing, storage, and management in the decentralized supply chain context;
- modified smart contracts to design, execute and enforce policy-based operations, e.g., data sharing and attestations;
- a secure and scalable access control approach to guarantee that only valid users can access a specific channel and the corresponding ledger, utilising TPM wallet to store the necessary certificates;
- an distributed Attribute-Based Encryption method based on keys securely stored in the TPMs of the engaged devices;
- a Searchable Encryption method operating over the public ledger used for secure data searching;
- an Off-Chain Storage Facility that is utilised to act as a secure storage container for each ASSURED deployment, storing the data in an encrypted state;
- a Secure Context Broker to resolve all requests and orchestrate the overall flow that needs to be executed over the ASSURED DLT network.

In the provided ASSURED DLT Architecture the placement and the description of the above-mentioned elements (alongside with others which are required by the ledger) is made evident, while this document also includes the documentation of the high-level flows which should be instantiated for enabling the communication between the different components.

All these will be further defined and design (in the sense of algorithms to be used and methods to be executes) in the consequent WP4 deliverables and the overall implementation and integration of the technical component will take place under WP5 of the project.

Funded by the Horizon 2020
Framework Programme of the European Union

ABBREVIATIONS

Abbreviation	Description
ABAC	Attribute-based Access Control
ABE	Attribute Based Encryption
AK	Attestation Key
API	Application Programming Interface
BFT	Byzantine Fault Tolerance
BGP	Byzantine Generals Problem
CA	Certification Authority
CP-ABE	Ciphertext Policy Attribute Based Encryption
CRED	AK Credential
DAA	Direct Anonymous Attestation
DApps	Distributed Applications
DLT	Distributed Ledger Technology
DoA	Description of Action
DPos	Delegated Proof of Stake
Dx.x	Deliverable x.xl
ECDSA	Elliptic Curve Digital Signature Algorithm
EK	Endorsement Key
HLF	Hyperledger Fabric
IoT	Internet of Things

Funded by the Horizon 2020 Framework Programme of the European Union

KDF	Key Derivation Function
KP-ABE	Key Policy Attribute Based Encryption
MSP	Membership Service Provider
PBFT	Practical Byzantine Fault Tolerance
PCR	Platform Configuration Register
PK	Public Key
PoA	proof of Authority
PoB	Proof of Burn
PoC	Proof of Capacity
PoET	Proof of Elapsed Time
PoS	Proof of Stake
PoW	Proof or Work
RA	Revocation Authority
SCB	Security Context Broker
SE	Searchable Encryption
SGX	Software Guard Extensions
SK	Secret Key
TPM	Trusted Platform Module
WPx	Work Package X

Funded by the Horizon 2020
Framework Programme of the European Union

REFERENCES

[1] Sivaganesan, D. "A Data Driven Trust Mechanism Based on Blockchain in IoT Sensor Networks for Detection and Mitigation of Attacks." Journal of trends in Computer Science and Smart technology (TCSST) 3.01 (2021): 59-69.

[2] ASSURED Consortium (2021). Deliverable D2.2 “Policy Modelling & Cybersecurity, Privacy and Trust Policy Constraints”.

[3] Hyperledger Fabric: a distributed operating system for permissioned Blockchains E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro and S. Muralidharan Proceedings of the Thirteenth EuroSys Conference, ACM (2018), p. 30

[4] Quorum Whitepaper, 2016. Available online: https://github.com/jpmorganchase/quorumdocs/blob/master/Quorum%20Whitepaper%20v0.1.pdf

[5] Brown RG. (2018). The corda platform: an introduction. Retrieved, 27, 2018.

[6] MultiChain Private Blockcchain - Whitepaper, 2015. Available online: https://www.multichain.com/download/MultiChain-White-Paper.pdf.

[7] Szabo, N., 1997. Formalizing and securing relationships on public networks. First monday.

[8] Ethereum white paper-a next generation smart contract and decentralized application platform-vitalik-buterin.pdf. http://Blockchainlab.com/pdf/Ethereum_white_paper-a_next_ generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf.

[9] Eliza Mik. Smart contracts: terminology, technical limitations and real world complexity. Law, Innovation and Technology, 9(2):269–300, 2017.

[10] Nakamoto, S. (2008) Bitcoin: A Peer-to-Peer Electronic Cash System, https://bitcoin.org/bitcoin.pdf.

[11] Vasin, P. (2014) Blackcoin’s Proof-of-Stake Protocol v2, https://blackcoin.co/blackcoin-pos-protocol- v2-whitepaper.pdf.

[12] Castro, M.; Liskov, B. Practical byzantine fault tolerance and proactive recovery. ACM Trans. Comput. Syst. 2002, 20, 398–461.

[13] Barinov, I.; Baranov, V.; Khahulin, P. POA Network Whitepaper; Technical Report; 2018. Available online: https://github.com/poanetwork/wiki/wiki/POA-Network-Whitepaper.

[14] Bistarelli, S.; Pannacci, C.; Santini, F. CapBAC in Hyperledger Sawtooth. In Distributed Applications and Interoperable Systems; Springer International Publishing: Cham, Switzerland, 2019; pp. 152–169.

[15] Dziembowski, Stefan; Faust, Sebastian; Kolmogorov, Vladimir; Pietrzak, Krzysztof (2015). Proofs of Space. Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference. 9216. pp. 585–605.

[16] Diego Ongaro, John K. Ousterhout: In Search of an Understandable Consensus Algorithm. USENIX Annual Technical Conference 2014: 305-319.

Funded by the Horizon 2020 Framework Programme of the European Union

[17] van Liesdonk, P., Sedghi, S., Doumen, J., Hartel, P., Jonker, W.: Computationally efficient searchable symmetric encryption. In: Jonker, W., Petković, M. (eds.) SDM 2010. LNCS, vol. 6358, pp. 87–100. Springer, Heidelberg (2010).

[18] Stefanov, E., Papamanthou, C., Shi, E.: Practical dynamic searchable encryption with small leakage. In: 21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA (2014).

[19] Bost, R.: ∑ οφος: forward secure searchable encryption. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS 2016, Vienna, Austria, pp. 1143–1154 (2016).

[20] Bost, R., Minaud, B., Ohrimenko, O.: Forward and backward private searchable encryption from constrained cryptographic primitives. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, pp. 1465–1482 (2017).

[21] Garg, S., Mohassel, P., Papamanthou, C.: TWORAM: efficient oblivious RAM in two rounds with applications to searchable encryption. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part III. LNCS, vol. 9816, pp. 563–592. Springer, Heidelberg (2016).

[22] Kim, K.S., Kim, M., Lee, D., Park, J.H., Kim, W.: Forward secure dynamic searchable symmetric encryption with efficient updates. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, pp. 1449–1463. ACM (2017).

[23] Song, X., Dong, C., Yuan, D., Xu, Q., Zhao, M.: Forward private searchable symmetric encryption with optimized I/O efficiency. IEEE Trans. Dependable Secur. Comput. 17(5), 912–927 (2020).

[24] Hoang, T., Yavuz, A.A., Guajardo, J.: Practical and secure dynamic searchable encryption via oblivious access on distributed data structure. In: Schwab, S., Robertson, W.K., Balzarotti, D. (eds.) Proceedings of the 32nd Annual Conference on Computer Security Applications, ACSAC 2016, Los Angeles, CA, USA, pp. 302–313. ACM (2016).

[25] Chamani, J.G., Papadopoulos, D., Papamanthou, C., Jalili, R.: New constructions for forward and backward private symmetric searchable encryption. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, pp. 1038–1055 (2018).

[26] Sun, S., et al.: Practical backward-secure searchable encryption from symmetric puncturable encryption. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, pp. 763–780 (2018).

[27] He, K., Chen, J., Zhou, Q., Du, R., Xiang, Y.: Secure dynamic searchable symmetric encryption with constant client storage cost. IEEE Trans. Inf. Forensics Secur. 16, 1538–1549 (2021).

[28] Demertzis, I., Chamani, J.G., Papadopoulos, D., Papamanthou, C.: Dynamic searchable encryption with small client storage. In: 27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA. The Internet Society (2020).

Funded by the Horizon 2020 Framework Programme of the European Union

[29] Amjad G., Kamara S., Moataz T.: Forward and backward private searchable encryption with SGX, in Proceedings of the 12th European Workshop on Systems Security, EuroSec 2019, Dresden, Germany, March 25, 2019, pp. 4:1–4:6.

[30] Sun, S., et al.: Practical non-interactive searchable encryption with forward and backward privacy. In: 28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, 21–25 February 2021. The Internet Society (2021).

[31] Amit Sahai, Brent Waters: Fuzzy Identity-Based Encryption. EUROCRYPT 2005: 457-473.

[32] Vipul Goyal, Omkant Pandey, Amit Sahai, Brent Waters: Attribute-based encryption for fine-grained access control of encrypted data. CCS 2006: 89-98.

[33] John Bethencourt, Amit Sahai, Brent Waters: Ciphertext-Policy Attribute-Based Encryption. IEEE Symposium on Security and Privacy 2007: 321-334.

[34] Jianting Ning, Zhenfu Cao, Xiaolei Dong, Kaitai Liang, Hui Ma, Lifei Wei: Auditable σ-Time Outsourced Attribute-Based Encryption for Access Control in Cloud Computing. IEEE Trans. Inf. Forensics Secur. 13(1): 94-105 (2018).

[35] Jinguang Han, Willy Susilo, Yi Mu, Jianying Zhou, Man Ho Allen Au: Improving Privacy and Security in Decentralized Ciphertext-Policy Attribute-Based Encryption. IEEE Trans. Inf. Forensics Secur. 10(3): 665-678 (2015).

[36] Zhen Liu, Zhenfu Cao, Duncan S. Wong: White-Box Traceable Ciphertext-Policy Attribute-Based Encryption Supporting Any Monotone Access Structures. IEEE Trans. Inf. Forensics Secur. 8(1): 76-88 (2013).

[37] Zhen Liu, Zhenfu Cao, Duncan S. Wong: Blackbox traceable CP-ABE: how to catch people leaking their keys by selling decryption devices on ebay. CCS 2013: 475-486.

[38] Jianting Ning, Zhenfu Cao, Xiaolei Dong, Lifei Wei, Xiaodong Lin: Large Universe Ciphertext-Policy Attribute-Based Encryption with White-Box Traceability. ESORICS (2) 2014: 55-72.

[39] Jianting Ning, Xiaolei Dong, Zhenfu Cao, Lifei Wei, Xiaodong Lin: White-Box Traceable Ciphertext-Policy Attribute-Based Encryption Supporting Flexible Attributes. IEEE Trans. Inf. Forensics Secur. 10(6): 1274-1288 (2015).

[40] Jianting Ning, Xiaolei Dong, Zhenfu Cao, Lifei Wei: Accountable Authority Ciphertext-Policy Attribute-Based Encryption with White-Box Traceability and Public Auditing in the Cloud. ESORICS (2) 2015: 270-289.

[41] Jianting Ning, Zhenfu Cao, Xiaolei Dong, Lifei Wei: White-Box Traceable CP-ABE for Cloud Storage Service: How to Catch People Leaking Their Access Credentials Effectively. IEEE Trans. Dependable Secur. Comput. 15(5): 883-897 (2018).

[42] Tatsuaki Okamoto, Katsuyuki Takashima: Fully Secure Unbounded Inner-Product and Attribute-Based Encryption. ASIACRYPT 2012: 349-366.

[43] Jie Chen, Romain Gay, Hoeteck Wee: Improved Dual System ABE in Prime-Order Groups via Predicate Encodings. EUROCRYPT (2) 2015: 595-624.

[44] Shashank Agrawal, Melissa Chase: FAME: Fast Attribute-based Message Encryption. CCS 2017: 665-682.

[45] Amit Sahai, Brent Waters: Fuzzy Identity-Based Encryption. EUROCRYPT 2005: 457-473.

Funded by the Horizon 2020 Framework Programme of the European Union

[46] Vipul Goyal, Omkant Pandey, Amit Sahai, Brent Waters: Attribute-based encryption for fine-grained access control of encrypted data. CCS 2006: 89-98.

[47] ASSURED Consortium (2021). Deliverable D1.4 “Report on Security, Privacy and Accountability Models”.

[48] Yao, Xuanxia, Zhi Chen, and Ye Tian. "A lightweight attribute-based encryption scheme for the Internet of Things." Future Generation Computer Systems 49 (2015): 104-112.

[49] Gayoso Martínez, Víctor, Luis Hernández Encinas, and Carmen Sánchez Ávila. "A survey of the elliptic curve integrated encryption scheme." (2010).

[50] Chen, Liqun, and Bogdan Warinschi. "Security of the TCG privacy-CA solution." 2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing. IEEE, 2010.

[51] The ASSURED Consortium, “D3.2 - ASSURED Layered Attestation and Runtime Verification Enablers Design and Implementation, November 2021.

[52] The ASSURED Consortium, “D1.4 – Report on Security, Privacy and Accountability Models for Dynamic Trusted Consent and Data Sharing”, August 2021.

[53] The ASSURED Consortium, “D2.2 – Policy Modelling & Cybersecurity, Privacy and Trust Policy Constraints”, November 2021.

[54] The ASSURED Consortium, “D1.2 – ASSURED Reference Architecture”, June 2021.

[55] The ASSURED Consortium, “D2.5 – Security Context Broker and Smart Contract Definition & Implementation for Policy Enforcement”, February 2022.

[56] The ASSURED Consortium, “D3.1 – ASSURED Attestation Model and Specification”, November 2021.

Funded by the Horizon 2020
Framework Programme of the European Union

APPENDIX A

A.1 A BYTECODE EXAMPLE

==========

PUSH1 0x80 PUSH1 0x40 MSTORE PUSH1 0x4 CALLDATASIZE LT PUSH2 0x41 JUMPI PUSH1 0x0 CALLDATALOAD PUSH29 0x100000000000000000000000000000000000000000000000000000000 SWAP1 DIV PUSH4 0xFFFFFFFF AND DUP1 PUSH4 0xCFAE3217 EQ PUSH2 0x46 JUMPI JUMPDEST PUSH1 0x0 DUP1 REVERT JUMPDEST CALLVALUE DUP1 ISZERO PUSH2 0x52 JUMPI PUSH1 0x0 DUP1 REVERT JUMPDEST POP PUSH2 0x5B PUSH2 0xD6 JUMP JUMPDEST PUSH1 0x40 MLOAD DUP1 DUP1 PUSH1 0x20 ADD DUP3 DUP2 SUB DUP3 MSTORE DUP4 DUP2 DUP2 MLOAD DUP2 MSTORE PUSH1 0x20 ADD SWAP2 POP DUP1 MLOAD SWAP1 PUSH1 0x20 ADD SWAP1 DUP1 DUP4 DUP4 PUSH1 0x0 JUMPDEST DUP4 DUP2 LT ISZERO PUSH2 0x9B JUMPI DUP1 DUP3 ADD MLOAD DUP2 DUP5 ADD MSTORE PUSH1 0x20 DUP2 ADD SWAP1 POP PUSH2 0x80 JUMP JUMPDEST POP POP POP POP SWAP1 POP SWAP1 DUP2 ADD SWAP1 PUSH1 0x1F AND DUP1 ISZERO PUSH2 0xC8 JUMPI DUP1 DUP3 SUB DUP1 MLOAD PUSH1 0x1 DUP4 PUSH1 0x20 SUB PUSH2 0x100 EXP SUB NOT AND DUP2 MSTORE PUSH1 0x20 ADD SWAP2 POP JUMPDEST POP SWAP3 POP POP POP PUSH1 0x40 MLOAD DUP1 SWAP2 SUB SWAP1 RETURN JUMPDEST PUSH1 0x60 PUSH1 0x40 DUP1 MLOAD SWAP1 DUP2 ADD PUSH1 0x40 MSTORE DUP1 PUSH1 0x5 DUP2 MSTORE PUSH1 0x20 ADD PUSH32 0x48656C6C6F000000000000000000000000000000000000000000000000000000 DUP2 MSTORE POP SWAP1 POP SWAP1 JUMP STOP LOG1 PUSH6 0x627A7A723058 KECCAK256 SLT 0xec 0xe 0xf5 0xf8 SLT 0xc7 0x2d STATICCALL ADDRESS SHR 0xdb COINBASE 0xb1 BALANCE 0xe8 0xf8 DUP14 0xda 0xad DUP13 LOG1 0x4c 0xb4 0x26 0xc2 DELEGATECALL PUSH7 0x8994D3E002900

==========

Funded by the Horizon 2020 Framework Programme of the European Union