ASSURE

ASSURED Blockchain based Secure Data Management

TU Delft

Kaitai Liang; Shihui Fu; Roland Kromes

Webinar

11st July 2023

www.project-assured.eu

Why do we need for ASSURED Blockchain

- 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**.
- 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.

Why do we need for ASSURED Blockchain

- Need for Distributed Ledger Technologies (DLTs) for Data Sharing
    - The usage of DLTs is to build a flexible and expandable data sharing network.
    - DLT offers smart contracts to govern the different data sharing activities that take place.
    - Smart contracts are used to govern the execution of attestations, clearly defining the flow of interactions between the Provers and the Verifiers.
    - DLT provides flexible onchain and offchain data storage.
- Need for Having DLTs to Facilitate ASSURED Trust Guarantees
    - DLTs have many other attractive features, such as decentralisation, persistence, anonymity, and auditability. These features are used by ASSURED to address the security, trust and confidentiality challenges which are present in an IoT SoS deployment.
- DLT combines with advanced access control mechanisms (e.g., ABAC) and data encryption (e.g., ABE).

Blockchain platform elements - HLF

Hyperledger Fabric (HLF), which is an **open-source** and one of the most popular **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.
- HLF can also support so-called **pluggable consensus** algorithms (e.g., PBFT, Raft10, Kafka11), 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 id entities are managed by the **membership service provider (MSP)**.

Blockchain platform elements - HLF

In the ASSURED, we 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 attribute-based encryption (**ABE)** and **searchable encryption**.

Blockchain platform elements - roles

- **Devices:** Entities that request to access smart contracts for the execution of attestation and for new enrolment verification.
- **Chaincode:** (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 is transferred via the form of transaction to peers who will install it on ledger, and then later be invoked by devices.
- **Orderer**. Given the smart contract execution responses (in the form of transactions) - which are sent by devices, orderers order and put them into a block.
- **Endorser.** This role is taken by the ASSURED blockchain network's peers. An endorser validates a given transaction with smart contract initialisation/call, and further execute the contract.
- **Security Context Broker (SCB)** is responsible for offering a trusted bridge between the (trusted) blockchain network and the (untrusted) "outside world". SCB interacts with smart contracts to 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 is also responsible for circulating (and enforcing) the secure enrolment policy.
- **Blockchain CA**: This is responsible for the verification of the device Blockchain credentials, in order to verify whether the device has permission to access the ledger in order to obtain the requested set of data.
- **Elastic Off-chain Storage Facility**: This is responsible for the storage of (raw) attestation data corresponding to an attestation report on the ledger, following the execution of an attestation operation. The attestation report stored on the ledger contains a location pointer that indicates the location of the stored data off-chain.

High level interactions: ASSURED Blockchain with other components

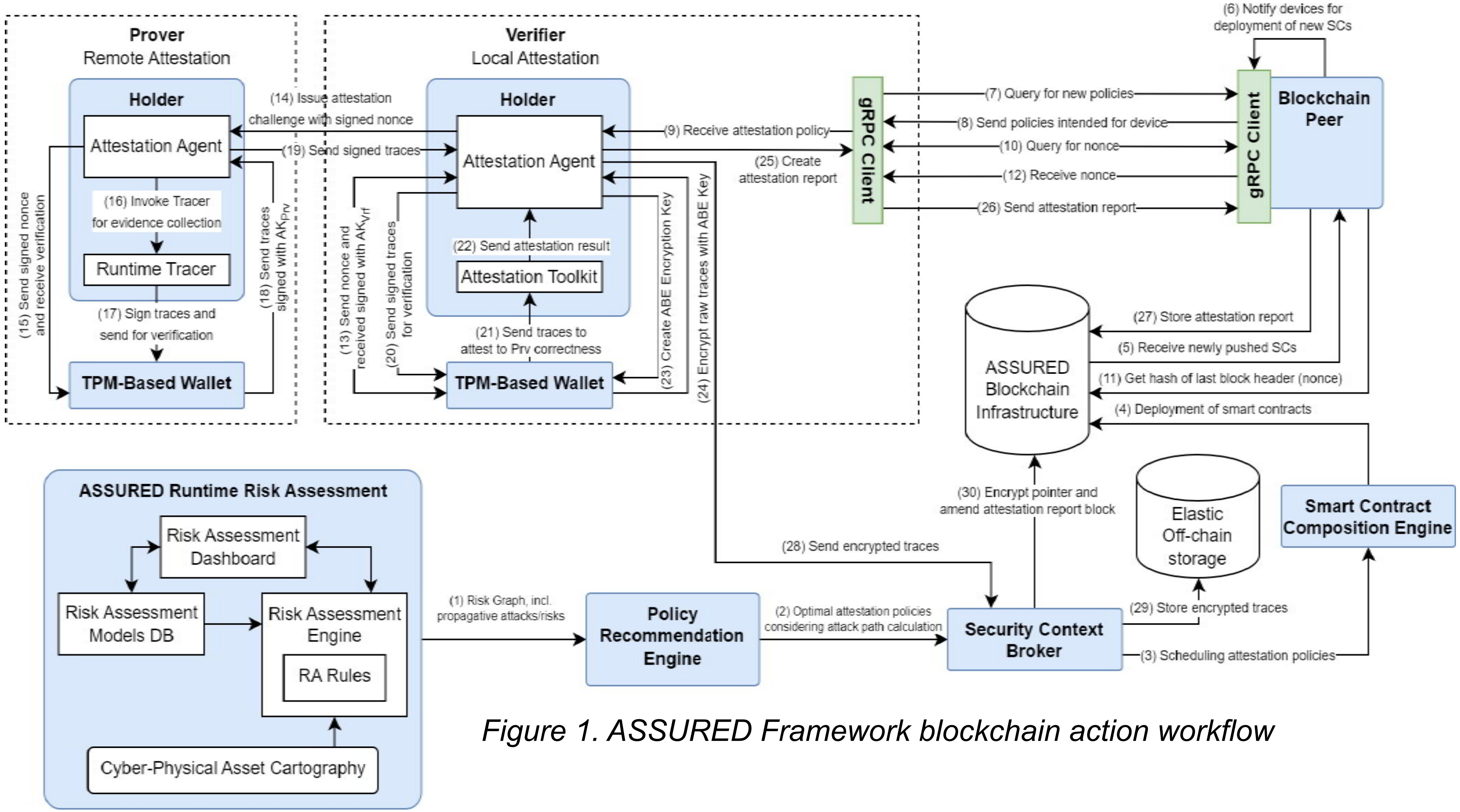

Prover
Remote Attestation
Holder
Attestation Agent
(16) Invoke Tracer for evidence collection
Runtime Tracer
(17) Sign traces and send for verification
TPM-Based Wallet
(15) Send signed nonce and receive verification
(18) Send traces signed with AK$_{Prv}$
(14) Issue attestation challenge with signed nonce
(19) Send signed traces
Verifier
Local Attestation
Holder
Attestation Agent
(22) Send attestation result
Attestation Toolkit
(21) Send traces to attest to Prv correctness
TPM-Based Wallet
(13) Send nonce and received signed with AK$_{Vrf}$
(20) Send signed traces for verification
(23) Create ABE Encryption Key
(24) Encrypt raw traces with ABE Key
(9) Receive attestation policy
(25) Create attestation report
gRPC Client
(7) Query for new policies
(8) Send policies intended for device
(10) Query for nonce
(12) Receive nonce
(26) Send attestation report
gRPC Client
Blockchain Peer
(6) Notify devices for deployment of new SCs
ASSURED Blockchain Infrastructure
(27) Store attestation report
(5) Receive newly pushed SCs
(11) Get hash of last block header (nonce)
(4) Deployment of smart contracts
(30) Encrypt pointer and amend attestation report block
(28) Send encrypted traces
Elastic Off-chain storage
Smart Contract Composition Engine
ASSURED Runtime Risk Assessment
Risk Assessment Dashboard
Risk Assessment Models DB
Risk Assessment Engine
RA Rules
Cyber-Physical Asset Cartography
(1) Risk Graph, incl. propagative attacks/risks
Policy Recommendation Engine
(2) Optimal attestation policies considering attack path calculation
Security Context Broker
(29) Store encrypted traces
(3) Scheduling attestation policies

Figure 1. ASSURED Framework blockchain action workflow

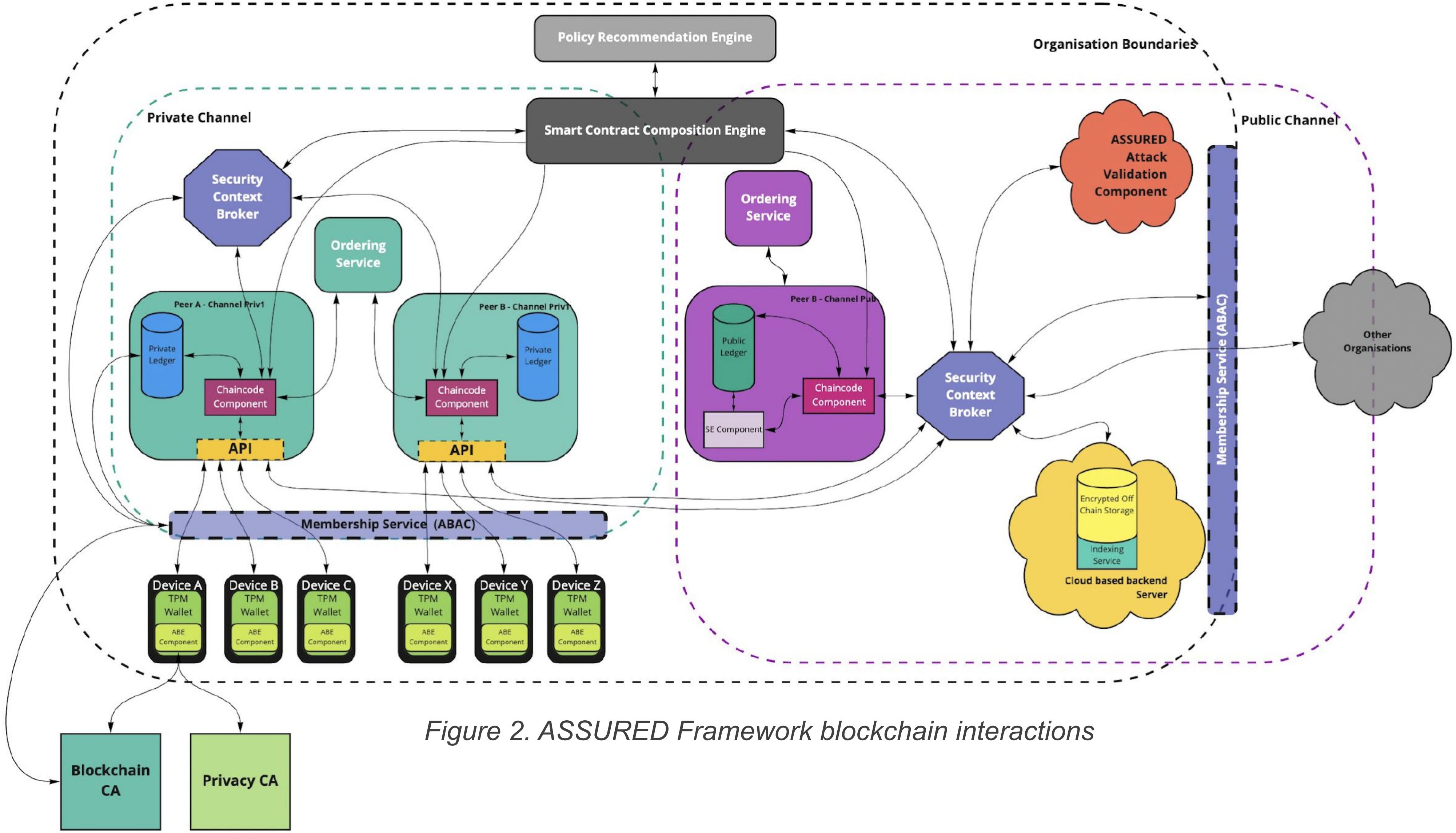


Figure 2. ASSURED Framework blockchain interactions

Blockchain platform elements - SC

Smart contract: event-driven computer programs defined, executed and enforced by the participants when a certain event happens, based on specified parameters in a blockchain network.

- It is first designed and deployed on the blockchain, and then a unique address is assigned to identify the smart contract.
- The specified blockchain users can invoke the smart contract by sending a transaction with the address.
- This transaction is later handled by an entity 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 is updated to the blockchain ledger.
- Only parties who have been granted permission can see the results/outputs.

Blockchain platform elements - SC

- We use smart contracts to provide **automatically and decentralized attestation operations and data accessing**.
- 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 is merged into a chaincode that enables attestation proof and verification, policy generation/update, and other data access operations.

Blockchain platform elements - SC

Types of SC:

- **Smart Contract depicting the scheduling attestation policies**
- This SC is based on the scheduling policies as outputted by the Policy Recommendation Engine for capturing the type of attestation schemes are executed by all edge devices towards achieving the required level of trustworthiness.
- This SC can provide all the necessary functions and information for a set of edge devices to execute a remote attestation process based on the attestation schemes.
- Input: 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.
- Output: 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.
- Interactions: Policy Recommendation Engine, Security Context Broker, Smart Contract Composition Engine, Edge devices.

Blockchain platform elements - SC

Types of SC:

- **Smart Contract depicting the querying of attestation related from external stakeholders**
- This SC can 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.
- Input: Access control policies depicting the list of attributes that the requesting stakeholders need to exhibit.
- Output: A transaction stored on the public ledger including the result of this data querying operation.
- Interactions: Security Context Broker, External Stakeholders.

Blockchain platform elements – SC

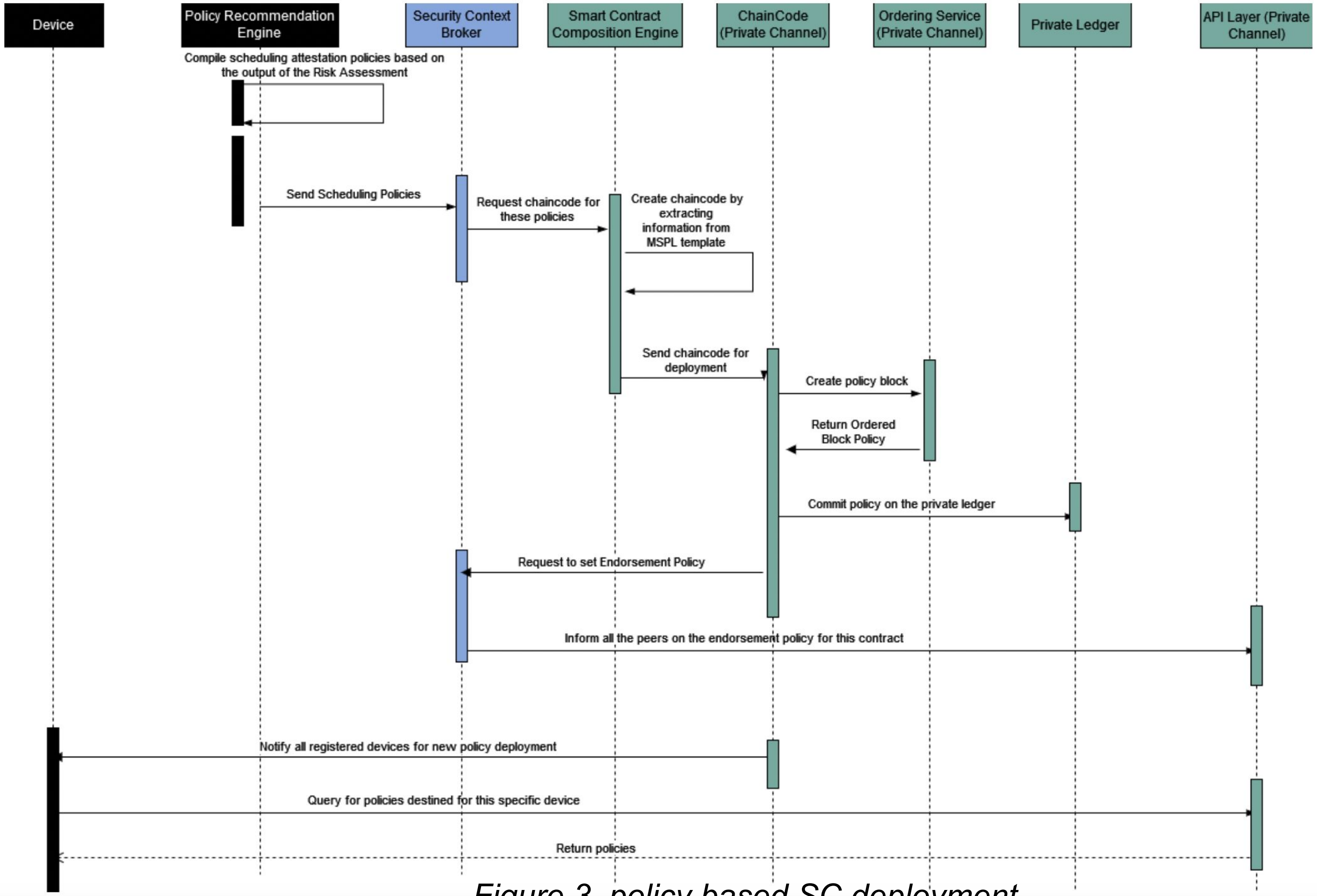

Device
Policy Recommendation Engine
Security Context Broker
Smart Contract Composition Engine
ChainCode (Private Channel)
Ordering Service (Private Channel)
Private Ledger
API Layer (Private Channel)
Compile scheduling attestation policies based on the output of the Risk Assessment
Send Scheduling Policies
Request chaincode for these policies
Create chaincode by extracting information from MSPL template
Send chaincode for deployment
Create policy block
Return Ordered Block Policy
Commit policy on the private ledger
Request to set Endorsement Policy
Inform all the peers on the endorsement policy for this contract
Notify all registered devices for new policy deployment
Query for policies destined for this specific device
Return policies

Figure 3. policy based SC deployment

Blockchain platform elements - consensus

Consensus.

- ***Raft*** is used 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 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.
- Raft is the only crash fault tolerant (instead of byzantine fault tolerant) consensus algorithm in the literature.

We use a fast and secure consensus algorithm to safeguard the final results from the validation and ordering stages.

This consensus algorithm is used in the ordering services so that a new block created by orderers could be finalized and validated.

Blockchain platform elements – Trusted Blockchain Wallet

ASSURED uses TPMs as central building block to build a very resource-efficient and trustful blockchain verification mechanism.

Trusted Authentication:

(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.

(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.

Blockchain platform elements – Trusted Blockchain Wallet

The ASSURED TPM-based Wallet offers several features to devices participating in the service graph chain towards achieving the requirements set forth by the ASSURED use case demonstrators.

Specifically, it offers identity management through the concept of Self-Sovereign Identities (SSI), which enables a device to make verifiable claims about its identity by leveraging its HW-based Trusted Component (TC). Thus, a device is able to transform its TPM-based Wallet into a trust anchor capable of securely managing Verifiable Credentials (VCs). These are essentially the set of device attributes that are issued by the Blockchain CA during the Secure Enrollment process, and include information, type of libraries installed, type of CPU microcontroller, as well as security claims on the correct execution of specific software.

The end goal of this approach is to achieve continuous authentication and authorization in the interactions of the devices with the Blockchain infrastructure, either when querying for operational or attestation-related data, or for recording attestation results. This is achieved by providing devices with the capability to create Verifiable Presentations (VPs), without the need for relying on trusted centralized entities for issuing them. Thus, when a device aims to perform a BC-related operation, it can create a VP based on the aforementioned VCs, containing only the subset of attributes required in order to perform that operation.

This approach aligns with the ASSURED goal of operating within a zero trust ecosystem, but also provides scalability by shifting trust to the devices themselves, thus eliminating the need for a centralized identity management system.

Blockchain platform elements - Trusted Blockchain Wallet

The main functionalities of the TPM wallet within ASSURED Blockchain services:

1. provide **strong user authentication** and **securely store the device credentials** based on the TPM's secure key storage functionality.
2. **control and authorize access to** private or public ledger channels based on the authentication process (e.g., to authorize access to or operations on different ledgers).
3. securely and efficiently **verify Blockchain updates**.
4. **continuously attest** to the security and trustworthiness of all involved devices in a privacy-preserving manner.

Blockchain platform elements - Trusted Blockchain Wallet

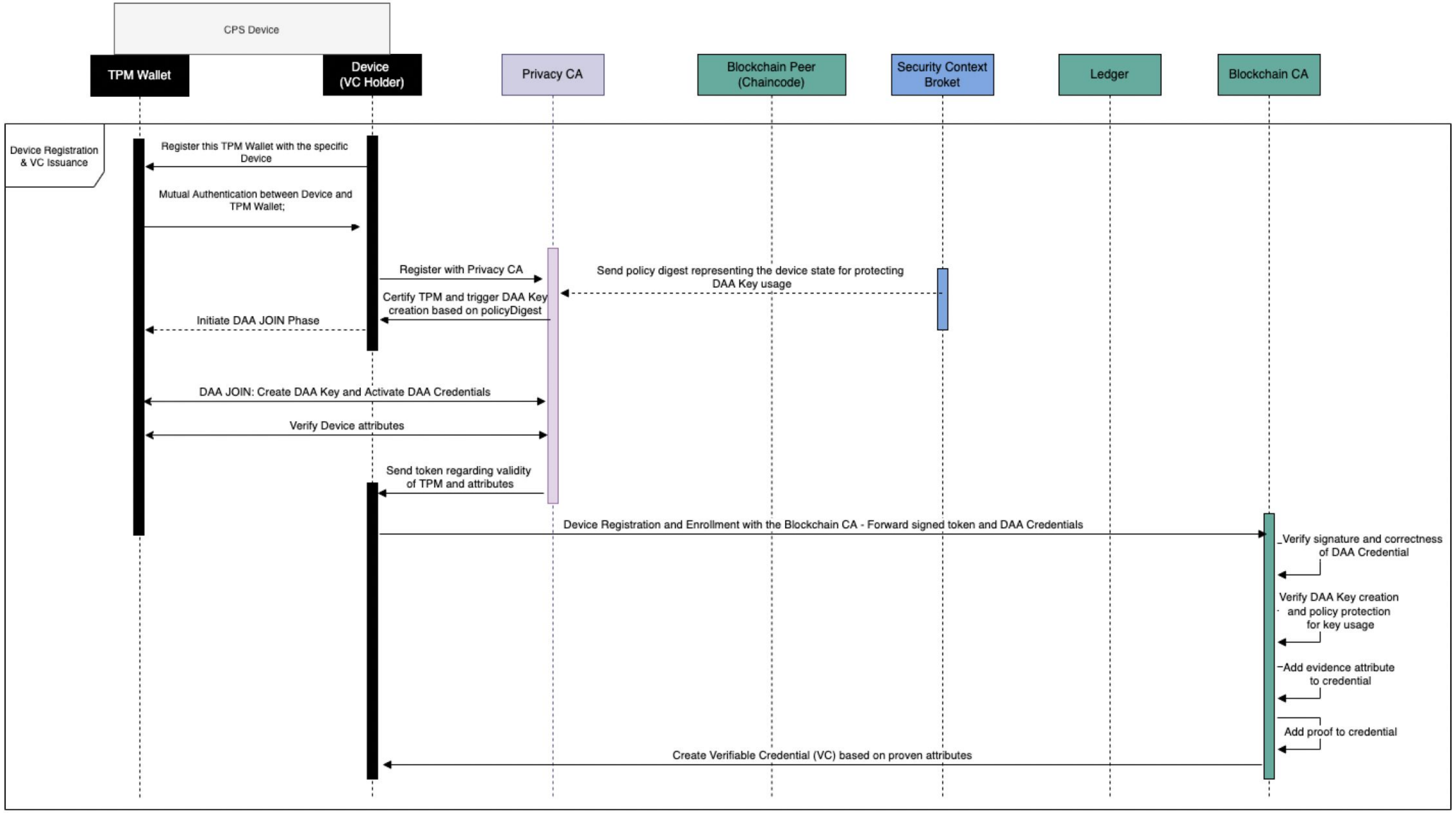
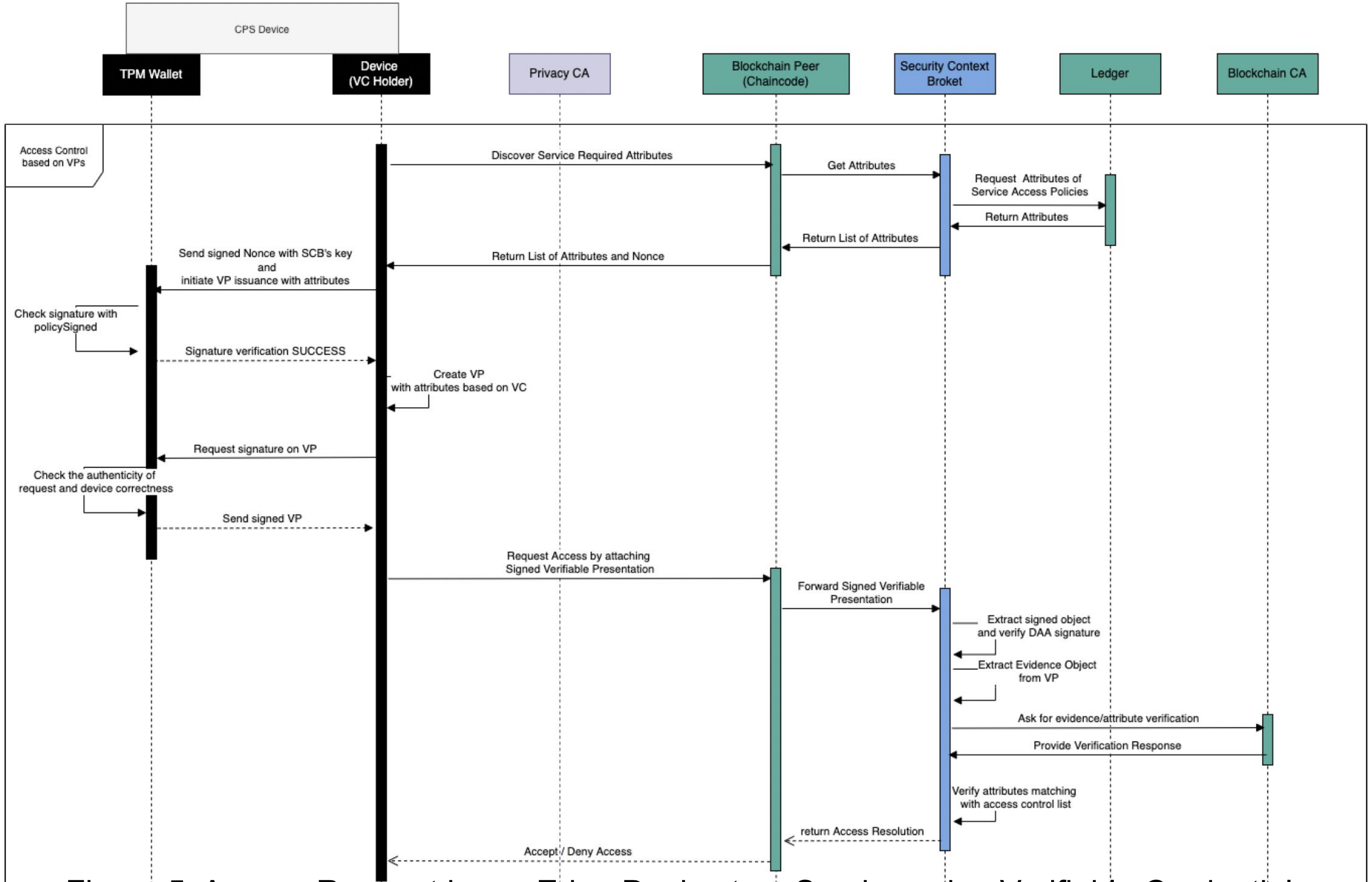

CPS Device
TPM Wallet
Device (VC Holder)
Privacy CA
Blockchain Peer (Chaincode)
Security Context Broket
Ledger
Blockchain CA
Device Registration & VC Issuance
Register this TPM Wallet with the specific Device
Mutual Authentication between Device and TPM Wallet;
Register with Privacy CA
Send policy digest representing the device state for protecting DAA Key usage
Certify TPM and trigger DAA Key creation based on policyDigest
Initiate DAA JOIN Phase
DAA JOIN: Create DAA Key and Activate DAA Credentials
Verify Device attributes
Send token regarding validity of TPM and attributes
Device Registration and Enrollment with the Blockchain CA - Forward signed token and DAA Credentials
Verify signature and correctness of DAA Credential
Verify DAA Key creation and policy protection for key usage
Add evidence attribute to credential
Add proof to credential
Create Verifiable Credential (VC) based on proven attributes

Figure 4. Registration of a Device in the ASSURED Blockchain

Blockchain platform elements - Trusted Blockchain Wallet



CPS Device

TPM Wallet

Device (VC Holder)

Privacy CA

Blockchain Peer (Chaincode)

Security Context Broket

Ledger

Blockchain CA

Access Control based on VPs

Discover Service Required Attributes

Get Attributes

Request Attributes of Service Access Policies

Return Attributes

Return List of Attributes

Return List of Attributes and Nonce

Send signed Nonce with SCB's key and initiate VP issuance with attributes

Check signature with policySigned

Signature verification SUCCESS

Create VP with attributes based on VC

Request signature on VP

Check the authenticity of request and device correctness

Send signed VP

Request Access by attaching Signed Verifiable Presentation

Forward Signed Verifiable Presentation

Extract signed object and verify DAA signature

Extract Evidence Object from VP

Ask for evidence/attribute verification

Provide Verification Response

Verify attributes matching with access control list

return Access Resolution

Accept / Deny Access

Figure 5. Access Request by an Edge Device to a Service using Verifiable Credentials

Blockchain platform elements - Trusted Blockchain Wallet

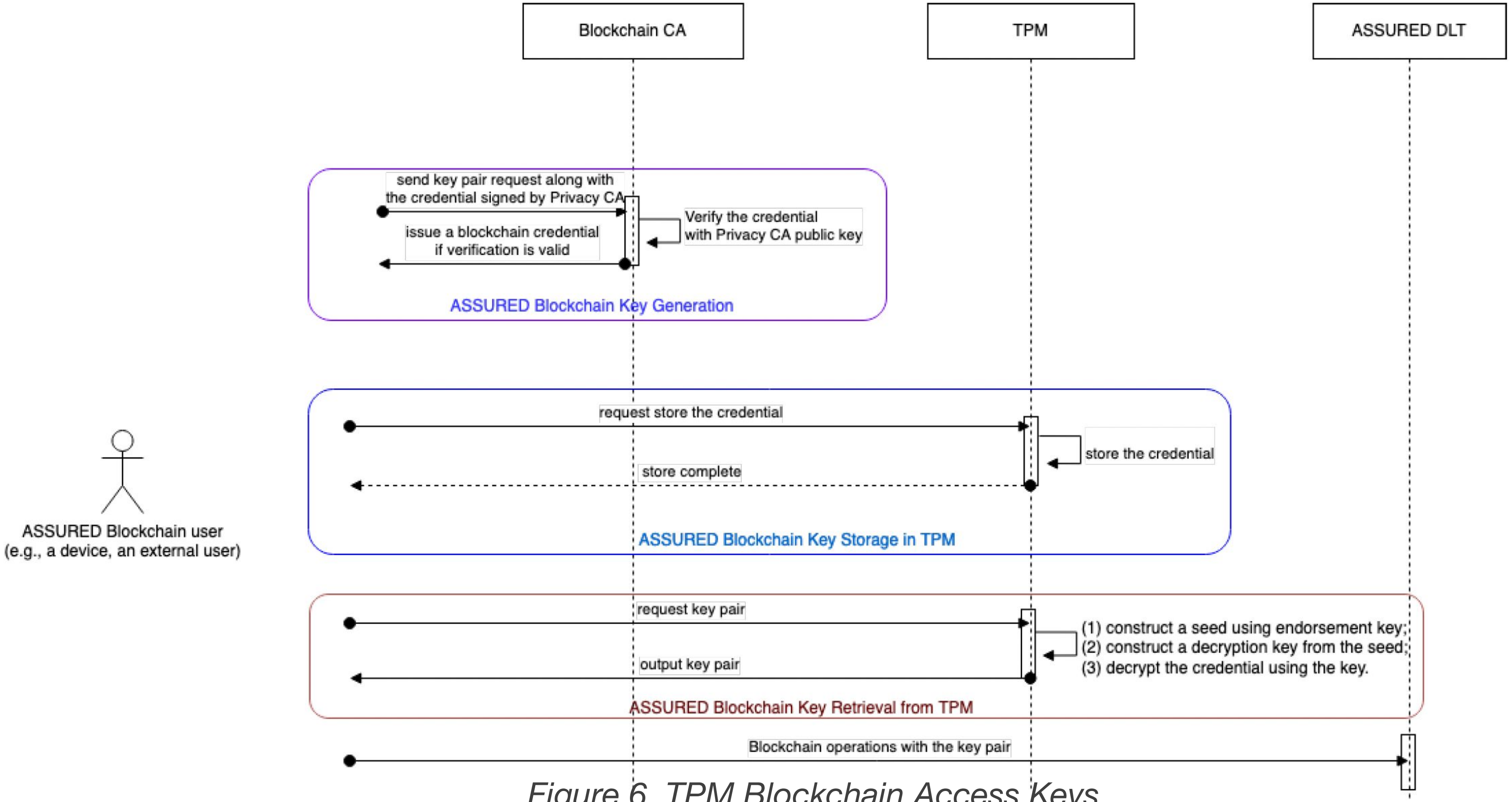

Blockchain CA
TPM
ASSURED DLT
send key pair request along with the credential signed by Privacy CA
Verify the credential with Privacy CA public key
issue a blockchain credential if verification is valid
ASSURED Blockchain Key Generation
request store the credential
store the credential
store complete
ASSURED Blockchain Key Storage in TPM
ASSURED Blockchain user (e.g., a device, an external user)
request key pair
(1) construct a seed using endorsement key;
(2) construct a decryption key from the seed;
(3) decrypt the credential using the key.
output key pair
ASSURED Blockchain Key Retrieval from TPM
Blockchain operations with the key pair

Figure 6. TPM Blockchain Access Keys

Secure data management

- Attribute-based access control (ABAC)
- Attribute-based encryption (ABE)
- Dynamic symmetric searchable encryption (DSSE)

Secure data management - ABAC

ABAC.

- ASSURED requires **secure access** on **private** and **public channels/ledgers** for all the network entities including devices, and the blockchain entities.
- Collaborations among **Privacy CA**, **Blockchain CA**, the **SCB** and the blockchain peers/orderers/clients.
  - The two CAs are used to generate token and credentials for devices.
  - SCB is regarded as a general role of admin to check devices’ policies and monitor the CAs operations.
  - Accessing blockchain channels and ledger must be verified via the devices’ credentials.
  - Any operations within the ASSURED blockchain network are signed by the entities’ credentials, so there are verifiers to check the validity of the signatures.

Secure data management - ABAC

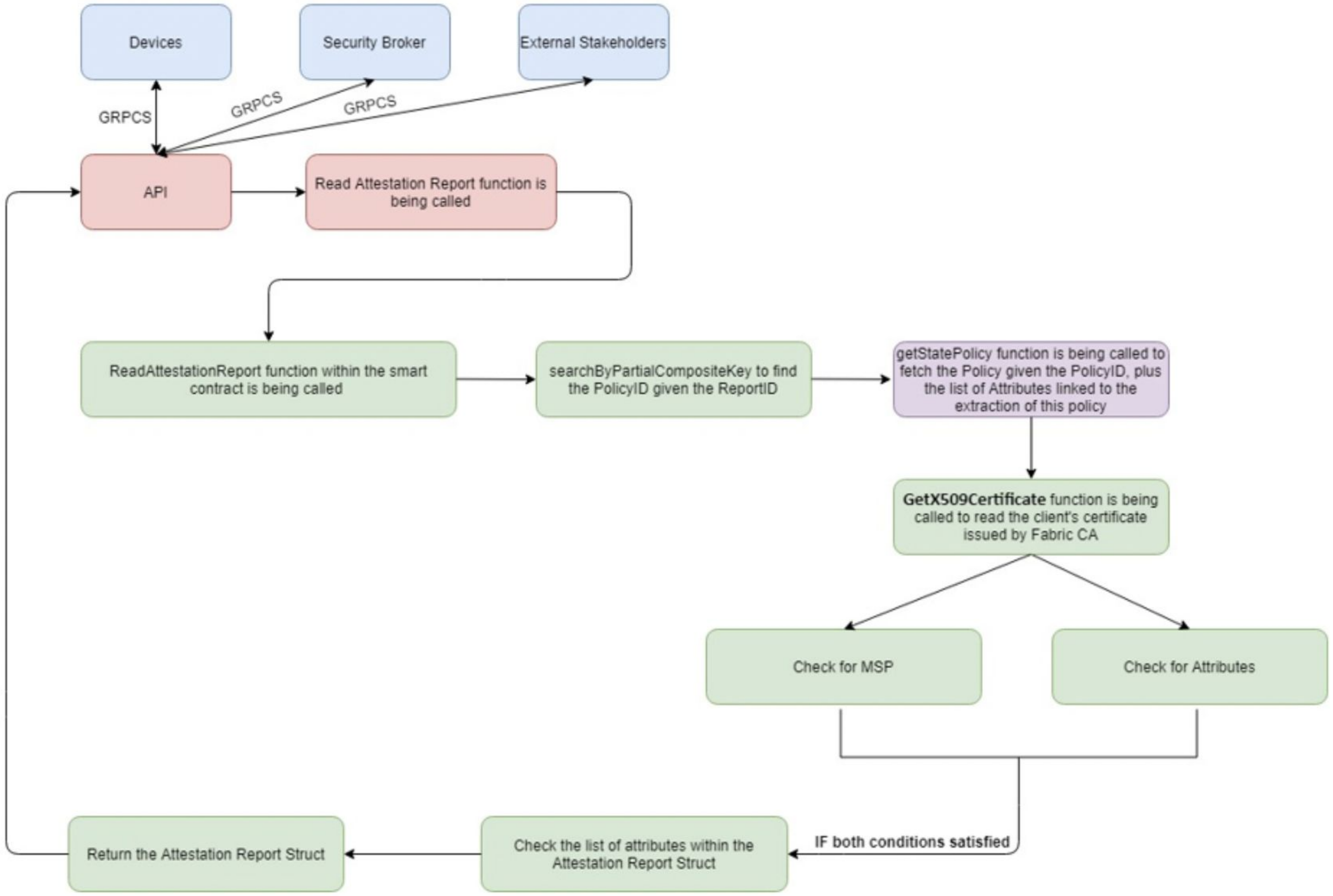

Devices
Security Broker
External Stakeholders
GRPCS
GRPCS
GRPCS
API
Read Attestation Report function is being called
ReadAttestationReport function within the smart contract is being called
searchByPartialCompositeKey to find the PolicyID given the ReportID
getStatePolicy function is being called to fetch the Policy given the PolicyID, plus the list of Attributes linked to the extraction of this policy
GetX509Certificate function is being called to read the client's certificate issued by Fabric CA
Check for MSP
Check for Attributes
IF both conditions satisfied
Check the list of attributes within the Attestation Report Struct
Return the Attestation Report Struct

Figure 7. General ABAC

Secure data management – ABAC

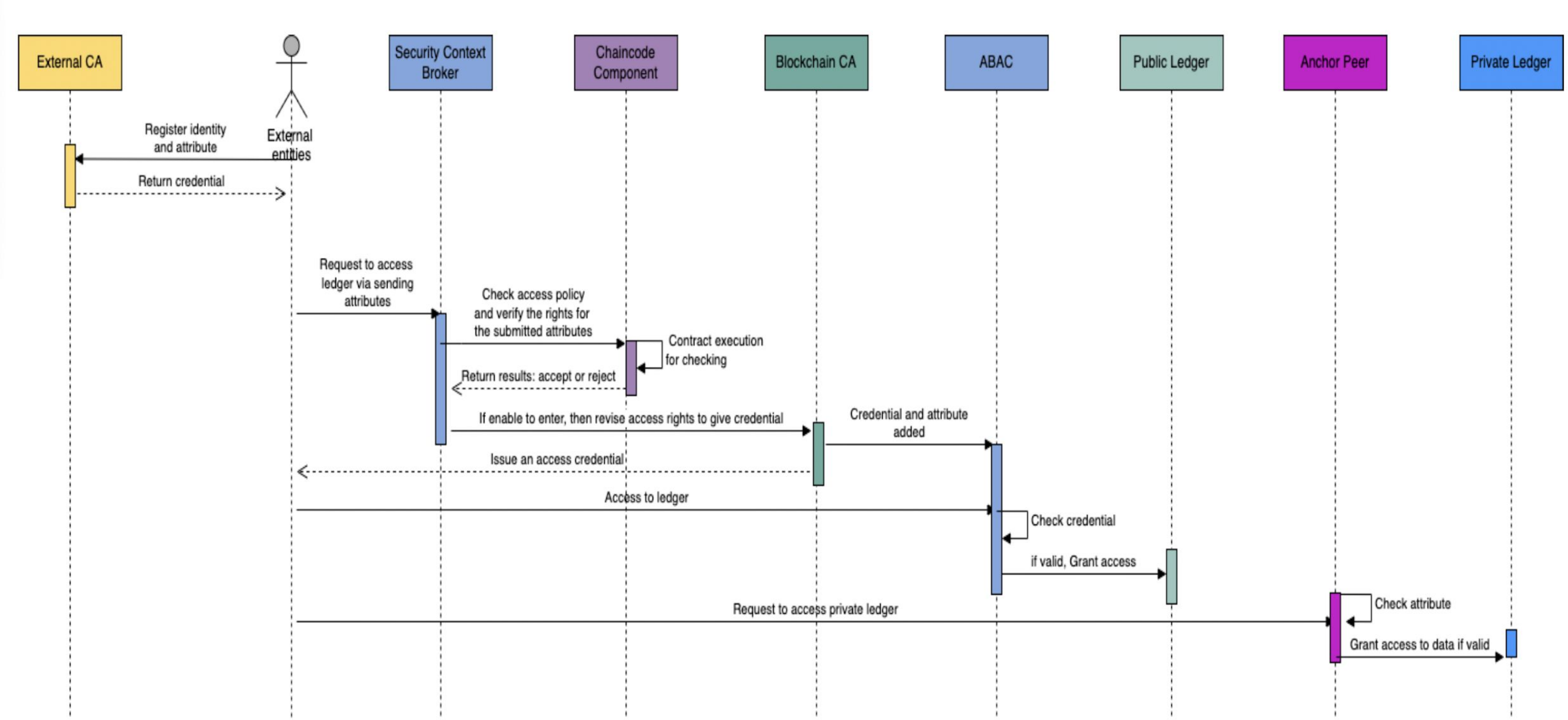

External CA
External entities
Security Context Broker
Chaincode Component
Blockchain CA
ABAC
Public Ledger
Anchor Peer
Private Ledger
Register identity and attribute
Return credential
Request to access ledger via sending attributes
Check access policy and verify the rights for the submitted attributes
Contract execution for checking
Return results: accept or reject
If enable to enter, then revise access rights to give credential
Credential and attribute added
Issue an access credential
Access to ledger
Check credential
if valid, Grant access
Request to access private ledger
Check attribute
Grant access to data if valid

Figure 8. ABAC: external entities channels access

Secure data management – ABAC

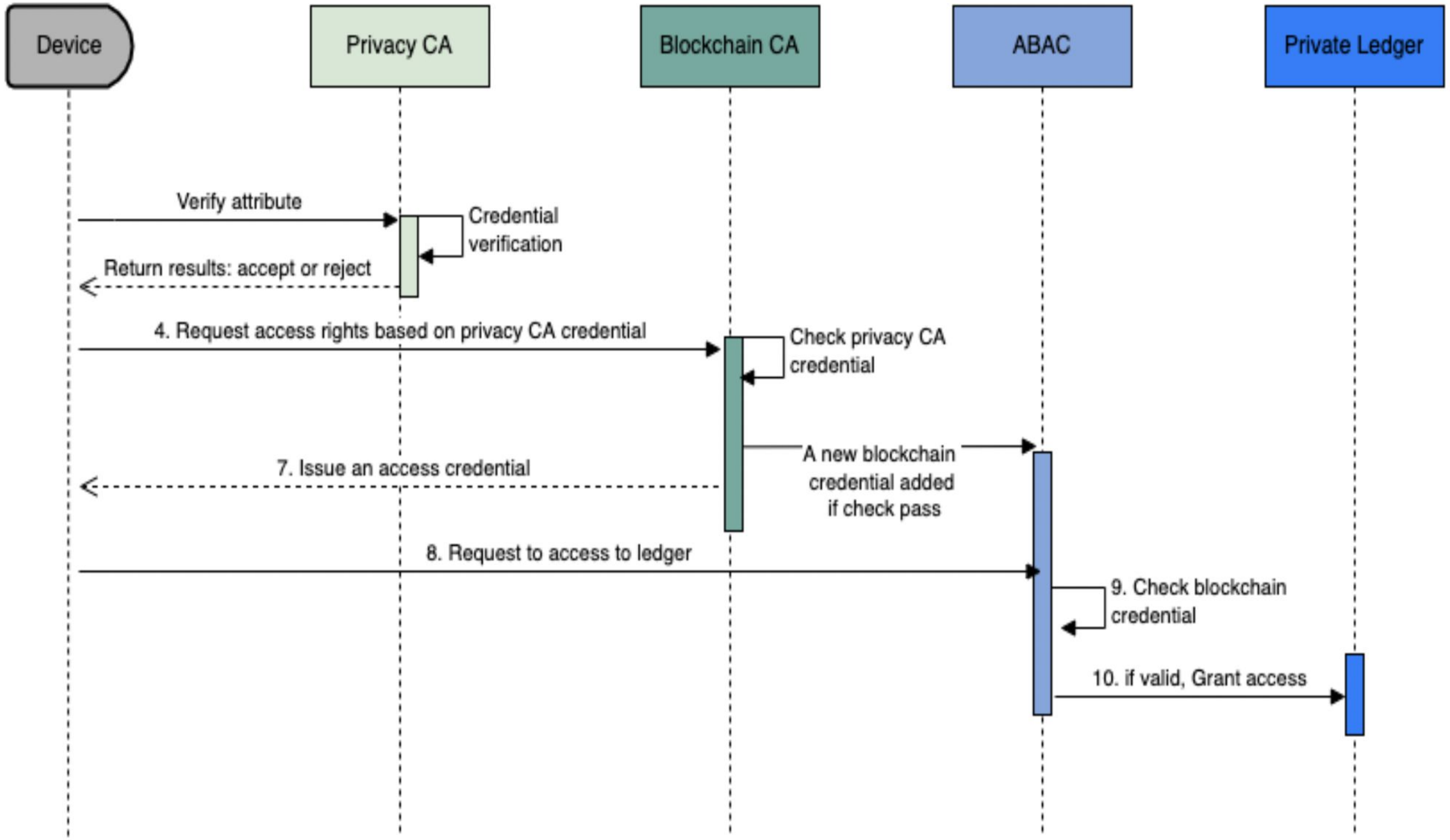

Device
Privacy CA
Blockchain CA
ABAC
Private Ledger
Verify attribute
Credential verification
Return results: accept or reject
4. Request access rights based on privacy CA credential
Check privacy CA credential
A new blockchain credential added if check pass
7. Issue an access credential
8. Request to access to ledger
9. Check blockchain credential
10. if valid, Grant access

Figure 9. ABAC: Private channels access

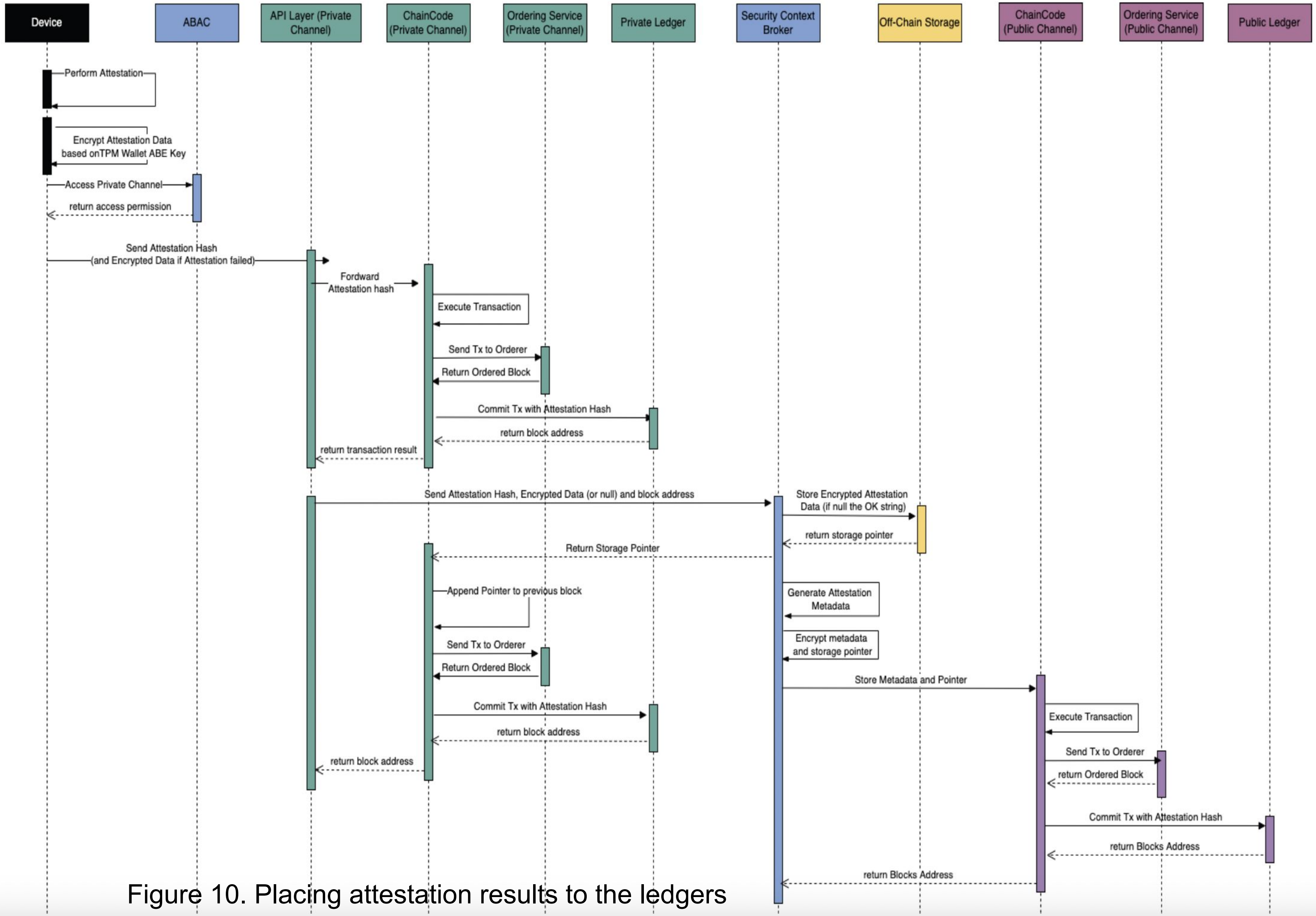

Device
ABAC
API Layer (Private Channel)
ChainCode (Private Channel)
Ordering Service (Private Channel)
Private Ledger
Security Context Broker
Off-Chain Storage
ChainCode (Public Channel)
Ordering Service (Public Channel)
Public Ledger
Perform Attestation
Encrypt Attestation Data based onTPM Wallet ABE Key
Access Private Channel
return access permission
Send Attestation Hash (and Encrypted Data if Attestation failed)
Fordward Attestation hash
Execute Transaction
Send Tx to Orderer
Return Ordered Block
Commit Tx with Attestation Hash
return block address
return transaction result
Send Attestation Hash, Encrypted Data (or null) and block address
Store Encrypted Attestation Data (if null the OK string)
return storage pointer
Return Storage Pointer
Append Pointer to previous block
Generate Attestation Metadata
Encrypt metadata and storage pointer
Send Tx to Orderer
Return Ordered Block
Store Metadata and Pointer
Commit Tx with Attestation Hash
return block address
Execute Transaction
return block address
Send Tx to Orderer
return Ordered Block
Commit Tx with Attestation Hash
return Blocks Address
return Blocks Address

Figure 10. Placing attestation results to the ledgers

Secure data management – data query

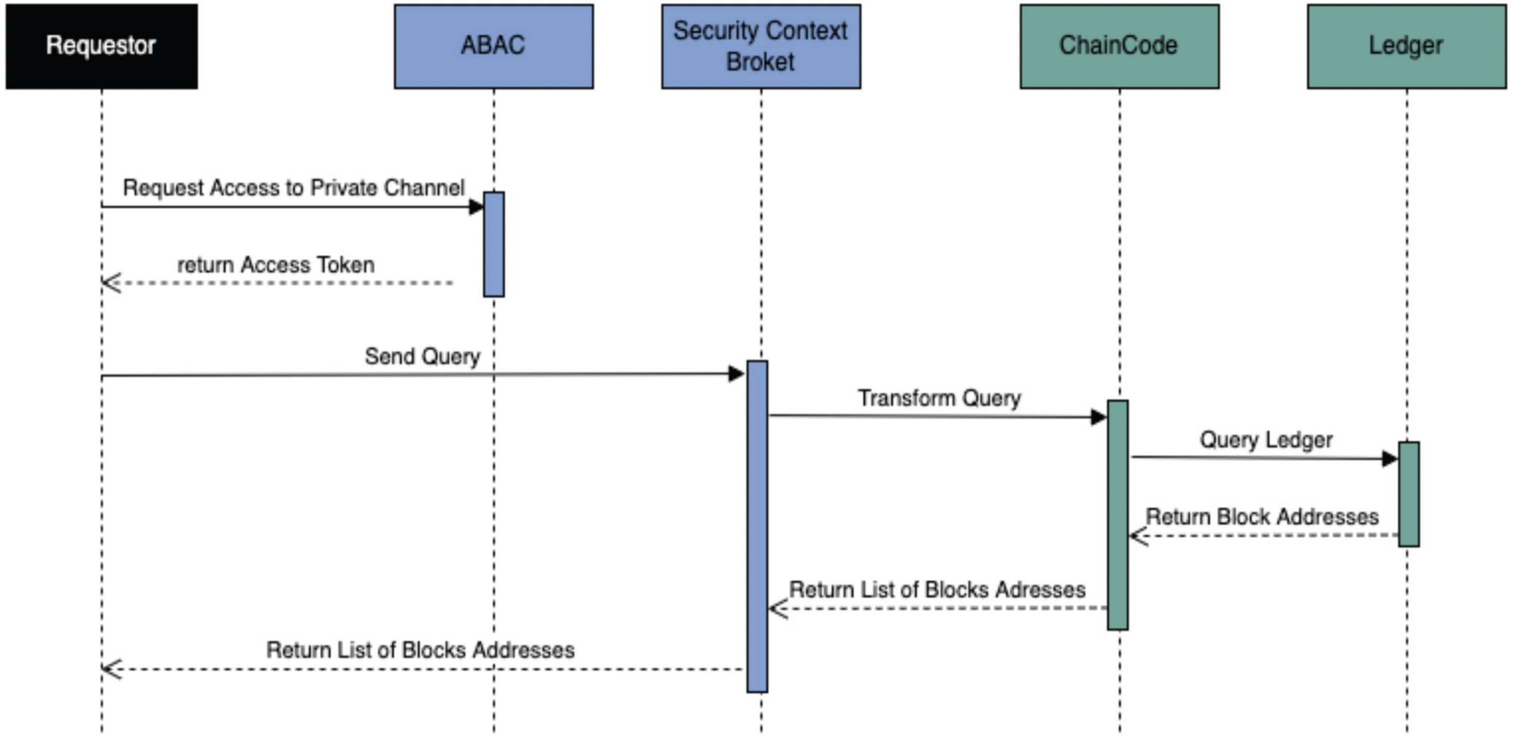


Figure 11. Private ledger query

Secure data management – attestation query from private ledger

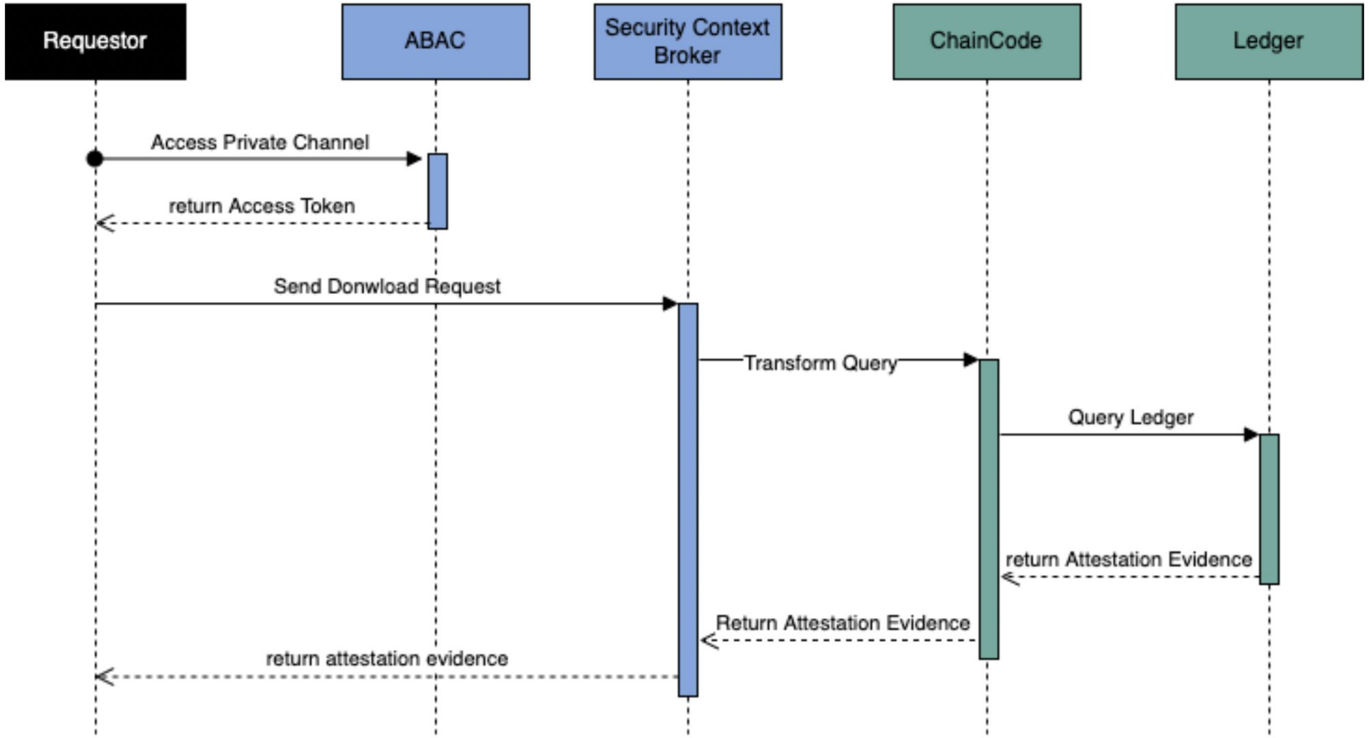


Figure 12. Extracting attestation evidence from the private ledger

Secure data management – data query

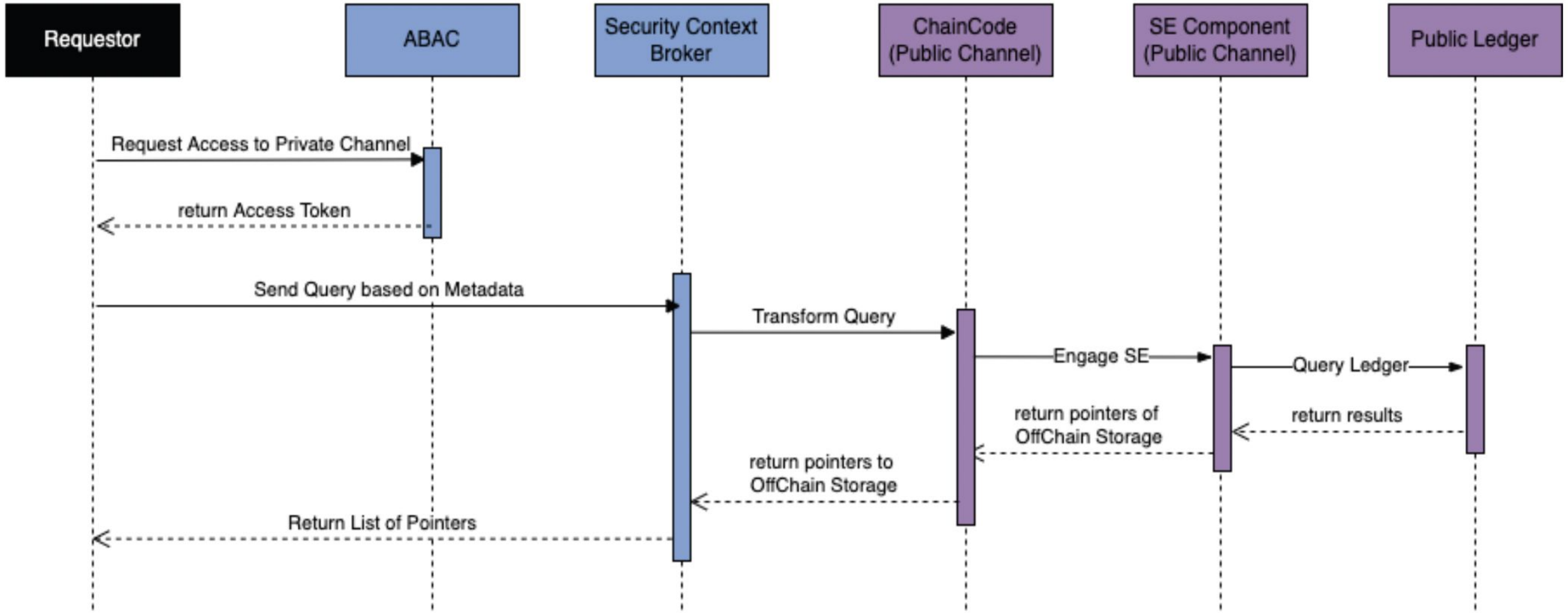

Requestor
ABAC
Security Context Broker
ChainCode (Public Channel)
SE Component (Public Channel)
Public Ledger
Request Access to Private Channel
return Access Token
Send Query based on Metadata
Transform Query
Engage SE
Query Ledger
return results
return pointers of OffChain Storage
return pointers to OffChain Storage
Return List of Pointers

Figure 13. Public ledger query

Secure data management– attestation raw data from off-chain storage

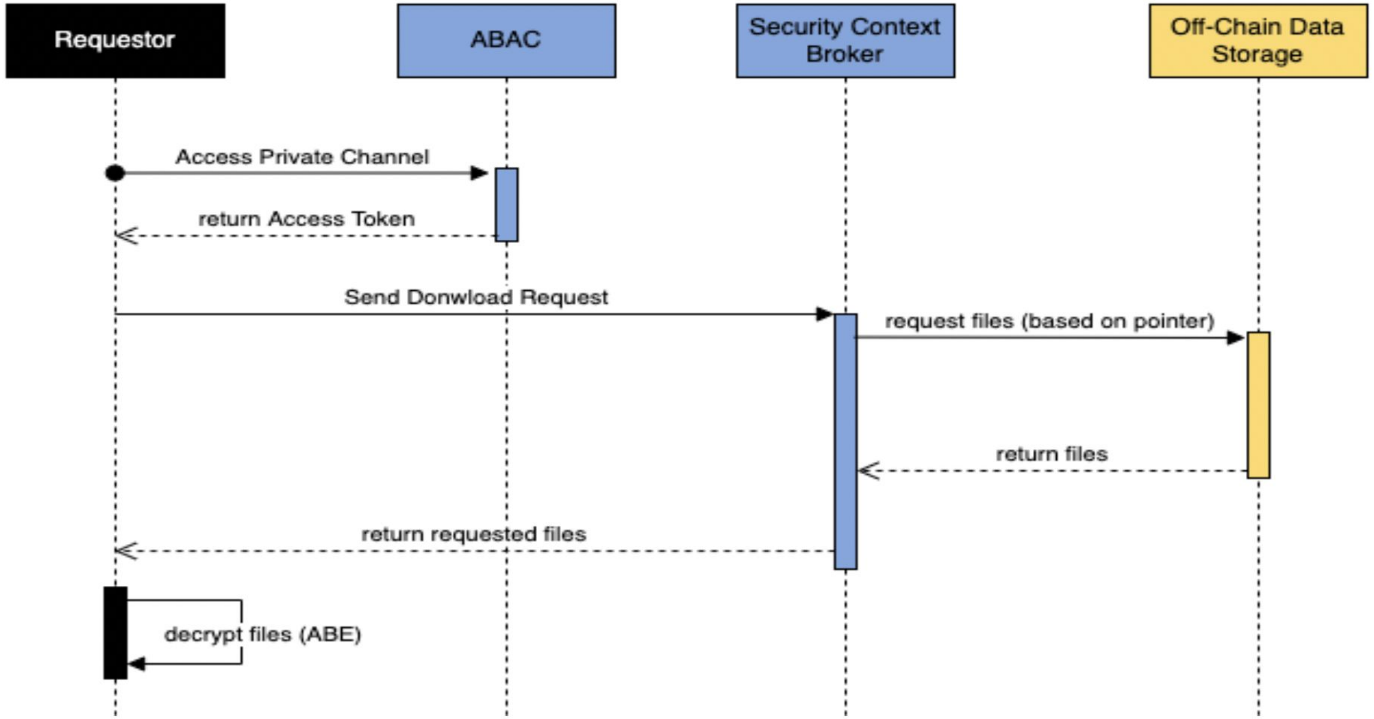


Figure 14. Extracting attestation results from the off-chain storage

Decentralized Attribute-based Encryption

- We 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.
- The concept of ABE offers a way to define asymmetric-key encryption schemes for policy enforcement based on attributes, where both the user secret key and the ciphertext are associated with a list of attributes.

Decentralized Attribute-based Encryption

ABE schemes that have been proposed in the literature operate in a *centralized* manner, that requires the presence of a trusted entity, referred to as the SCB, to whom the devices send their raw attestation data.

This entity not only dictates which are the attribute policies but also is responsible for managing the encryption and decryption keys, and for encrypting and decrypting the data depending on who performs the request.

However, the issue with this approach is that a very large degree of trust is placed on the SCB, since it has access to the actual data, as well as the cryptographic keys.

The novelty of ASSURED in this regard is that we propose the *first scheme of its kind* to provide **decentralized ABE**, that enables encryption and decryption based on the presence and verification of specific attributes in the requesting devices.

Specifically, we enable the encryption process at the side of the data owner, so there is no need for a central entity such as the SCB to have access to the actual data or the keys.

Decentralized Attribute-based Encryption

System model:

- **Trusted Authority:** responsible for defining the attribute and generating the user attribute keys for each of these attributes. In ASSURED, the SCB is the party acting as the trusted authority.
- **Encryptor:** The device that aims to encrypt data in a manner that makes decryption possible only by parties that possess the appropriate attributes, which are specified during encryption. The encrypted data may be attestation data which is generated after the execution of an attestation process and needs to be recorded to the ledger and made accessible to interested parties (mainly in the BIBA Smart Manufacturing, UTRC Smart Aerospace, and SPH Smart Satellites use cases), or operational data which may be extracted during the operation of devices such as smoke sensors (mainly in the DAEM Smart Cities use case).
- **Decryptor:** The device that aims to decrypt data that has been encrypted by the Encryptor. The Decryptor must possess the attributes specified by the Encryptor during the encryption process. Note that the totality of these attributes has been registered during the Secure Enrollment process in the form of Verifiable Credentials (VCs), and in order to access a set of encrypted data, the Decryptor must create a Verifiable Presentation (VP) containing only the subset of attributes required in order to perform the decryption.

Decentralized Attribute-based Encryption

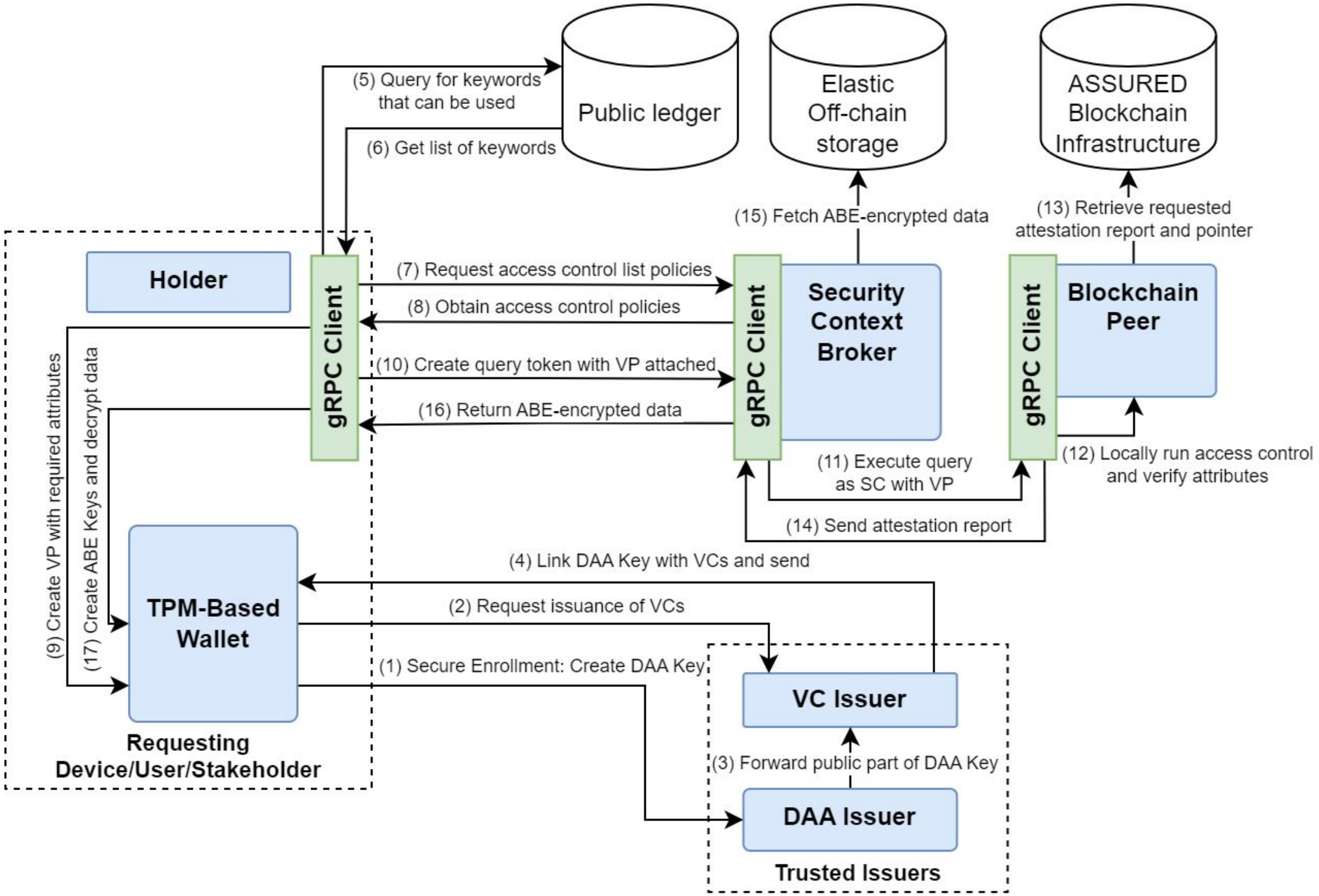

Public ledger
Elastic Off-chain storage
ASSURED Blockchain Infrastructure
(5) Query for keywords that can be used
(6) Get list of keywords
(15) Fetch ABE-encrypted data
(13) Retrieve requested attestation report and pointer
Holder
gRPC Client
(7) Request access control list policies
(8) Obtain access control policies
(10) Create query token with VP attached
(16) Return ABE-encrypted data
gRPC Client
Security Context Broker
gRPC Client
Blockchain Peer
(12) Locally run access control and verify attributes
(11) Execute query as SC with VP
(14) Send attestation report
(9) Create VP with required attributes
(17) Create ABE Keys and decrypt data
TPM-Based Wallet
(4) Link DAA Key with VCs and send
(2) Request issuance of VCs
(1) Secure Enrollment: Create DAA Key
VC Issuer
(3) Forward public part of DAA Key
DAA Issuer
Trusted Issuers
Requesting Device/User/Stakeholder

Figure 15: Querying for ABE-encrypted data in ASSURED

Decentralized Attribute-based Encryption

Conceptual Protocol Overview

Initialization phase:

1. Either as an Encryptor or as a Decryptor. First, the device sends its static attributes (which will eventually be used as part of Verifiable Presentations) to the SCB.
2. The SCB creates the static attribute elliptic keys, along with a key for the device to put into its key hierarchy in order to create the encryption/decryption and HMAC keys under a key policy, containing a concatenation of the static attributes and the dynamic attributes.
3. The data is encrypted under the device (TPM)'s endorsement key.
4. From the side of the device, when the encrypted data is acquired, it decrypts and stores the attribute keys inside the PCRs of its embedded TPM, and puts the key provided by the SCB in the TPM's key hierarchy.

Decentralized Attribute-based Encryption

Conceptual Protocol Overview

Encryption phase:

1. Encryptor first requests from its TPM to create a random nonce.
2. It uses this nonce to compute, along with the master attribute key and the key restriction policy, the encryption key seed.
3. If then creates the Encryption Key under the key provided by the SCB, bonded with the appropriate policy.
4. The device undergos an integrity check, using both its static attributes and its dynamic attributes.
5. If the device passes the integrity check, it encrypts the desired data and creates its HMAC.
6. Then, the device computes a shared value, using the Public static attribute keys, and the aforementioned random nonce created by the TPM.
7. The device then uses its DAA key that was created during the Secure Enrollment, to perform a DAA signature operation on the key restriction policy, to provide a security proof that the correct key restriction policy was used during the encryption of the raw traces.
8. The device sends the encrypted data, the HMAC of the encrypted data, the signature of the policy, the public part of the signing key, its respective credential, and the shared value, to the SCB in order to be stored in the data storage engine.
9. The signature of the policy and the public part of the signing key, are stored in the ledger.

Decentralized Attribute-based Encryption

Conceptual Protocol Overview

Decryption phase:

1. Decryptor first retrieves the DAA-signature.
2. It uses the policy that was expected to be used in order to verify the signature provided by the Encryptor and to recreate the credential.
3. If this verification is successful, the Decryptor uses its static attribute key to compute the encryption key seed.
4. With the extracted key seed, it creates an HMAC key in order to recompute the HMAC of the encrypted data.
5. It uses this HMAC in order to perform an authentication check, by comparing it to the one given by the Encryptor.
6. If the authentication check is successful, the device goes through an integrity check, using both the static and dynamic attributes.
7. Upon successful completion of the integrity check, the Decryptor device decrypts the data.

Decentralized Attribute-based Encryption

ASSURED

Attribute list update

1. SCB requests from the device a session nonce.
2. Using the nonce, SCB signs the new policy that contains the concatenation of the run-time and the static attributes.
3. It sends the new signed policy to the corresponding device, along with the corresponding cpHash.
4. The device then verifies the signature.
5. If the verification is successful, the old policy is replaced with the new one.

Dynamic Searchable Symmetric Encryption

DSSE.

- ASSURED requires that any external parties can perform **fast** and **secure search** over the public ledger to locate any related attestation files.

Dynamic Searchable Symmetric Encryption

Need of Searchable Encryption

Only 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 in ASSURED, 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. The off-chain data is stored over a cloud-based data storage engine which may not be trusted with security of sensitive data. As known, data privacy issue are the most prominent and important one when it comes to cloud storage, and could theoretically be easily resolved by storing the data in an encrypted form. However, although encryption solves the problem of privacy, it also engenders some other serious issues including infeasibility of the fundamental search operation, reduction in flexibility of sharing the data with other users etc. To address these issues, the concept of searchable encryption is introduced and developed on ASSURED blockchain ledger. SE allows blockchain ledger peer(s) to perform search on encrypted and stored data on ledger(s) without disclosing any information about what is being searched to the peer(s). Secure SE is the answer to this need.

Dynamic Searchable Symmetric Encryption

DSSE.
This service mainly includes the interactions with SCB, the ASSURED public ledger, ASSURED data storage engine and the searchable encryption component.

DSSE component can 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 are provided by the SCB in **an encrypted manner**, and this component allows stakeholders to request 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 is revealed to the query party, reverting them to the storage engine where they should query to get the full attestation report.

We consider using this as a **dynamic** mode to enable the increase of metadata from various attestation reports.

Dynamic Searchable Symmetric Encryption

- **Data Privacy.** DSSE depends on a third-party server to receive and store scripts which imply a keyword-document relationship for later data retrieval. That is, the index or the searchable ciphertext is visible to the server. Data privacy requires that the content of documents should be concealed, even though index or searchable ciphertext is transmitted and stored.

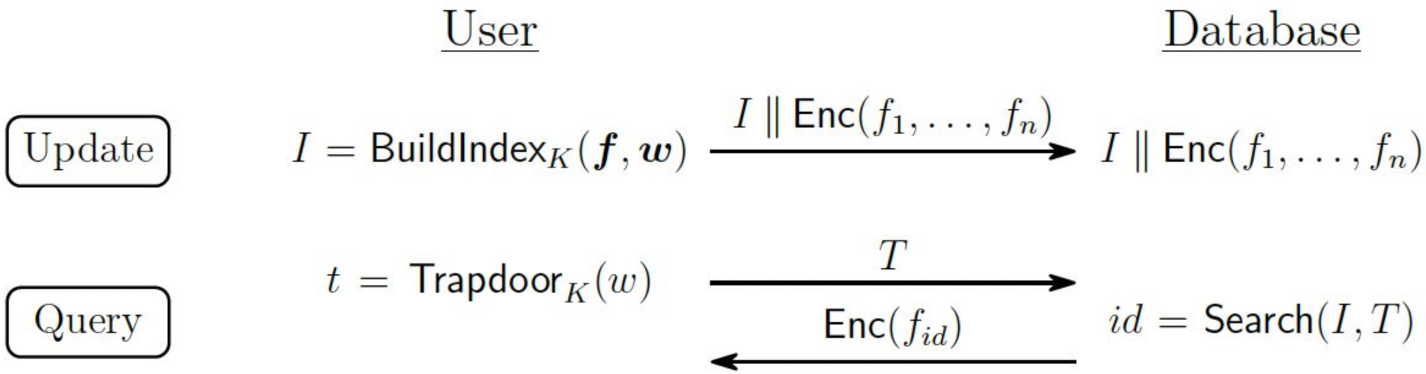
- **Keyword Privacy.** Given a search query, i.e., the token or the trapdoor representing the keywords of user’s interest, keyword privacy requires that the token or the trapdoor should not leak information about the underlying keyword while used for testing index or searchable ciphertexts.

Dynamic Searchable Symmetric Encryption

DSSE includes four algorithms:

- **Setup/Key Generation.** In this algorithm, system parameters are set, and keys are generated for each party.
- **Build Index/Encrypt.** In this algorithm, the data owner processes the keywords of their documents and uploads this script, i.e., index or searchable ciphertext, to the server.
- **Token/Trapdoor Generation.** In this algorithm, the authorized data user computes the token or trapdoor based on the keyword of its interest, which is used for later testing with an index item or a searchable ciphertext.
- **Search/Test.** In this algorithm, the server validates the received token or trapdoor with the index item or searchable ciphertext in storage to decide to return the corresponding document or not.

Dynamic Searchable Symmetric Encryption



User

Database

Update

$I = \mathsf{BuildIndex}_K(\boldsymbol{f}, \boldsymbol{w})$

$I \parallel \mathsf{Enc}(f_1, \ldots, f_n)$

$I \parallel \mathsf{Enc}(f_1, \ldots, f_n)$

Query

$t = \mathsf{Trapdoor}_K(w)$

T

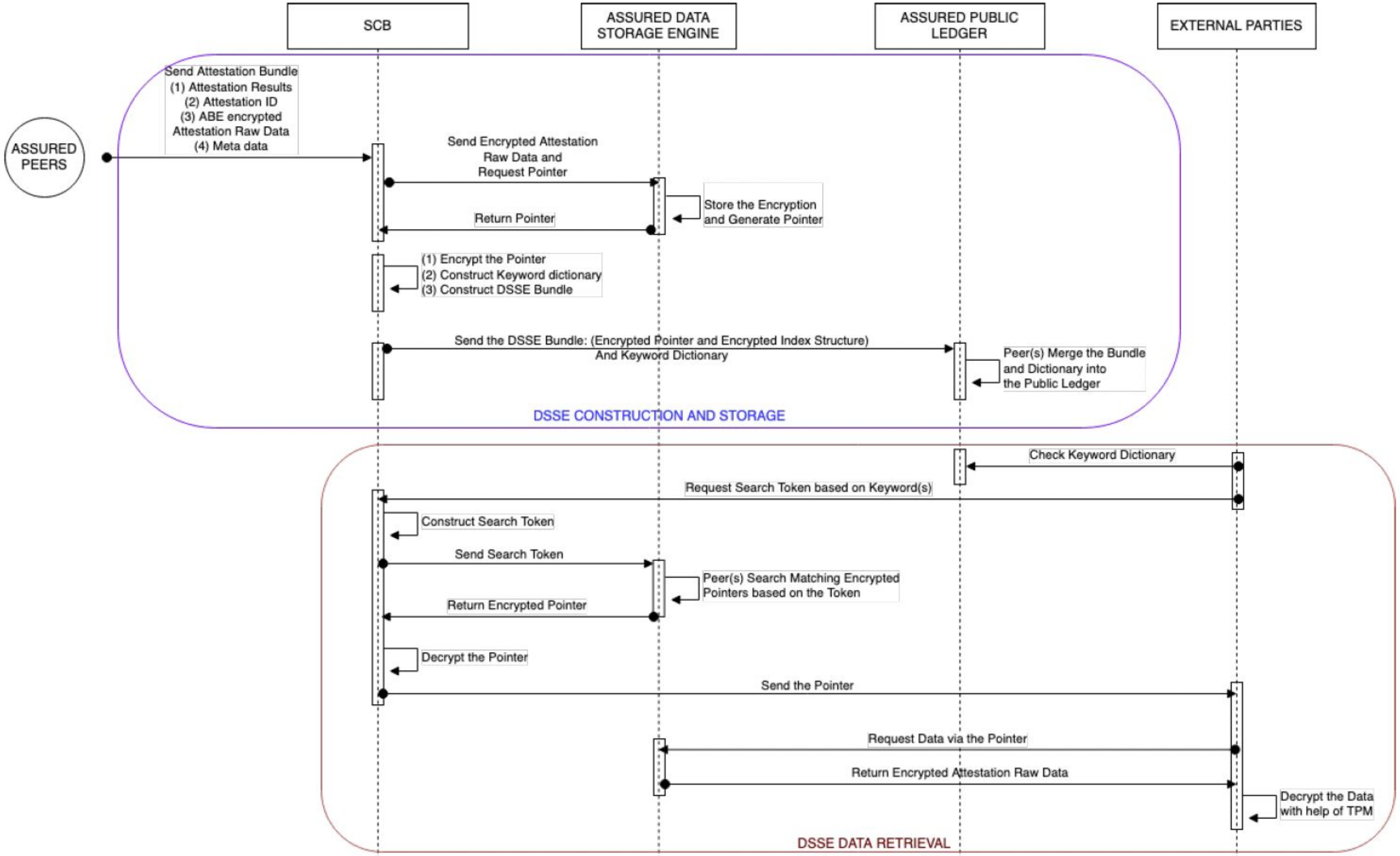
$\mathsf{Enc}(f_{id})$

$id = \mathsf{Search}(I, T)$

Figure 16. General Model of an Index-Based Searchable Encryption Scheme

Dynamic Searchable Symmetric Encryption

- **Data owner:** An authorized data owner outsources a collection of documents f = (f_1, . . . , f_n) together with some keywords w = (w_1, . . . ,w_m). In ASSURED context, the devices who generate the attestation data are the data owners.
- **Blockchain peers:** send the attestation bundle, including attestation results, attestation ID, encrypted attestation raw data, and related metadata to the SCB; perform secure share over the public ledger.
- **ASSURED public ledgers:** The public ledger stores the encrypted index structure and encrypted pointers (Recall the private ledger is used to store attestation results and related information).
- **SCB**: It makes use of searchable encryption component to construct encrypted index structure and encrypted pointer, and further stores them on the public ledger. The SCB also performs search/update token generation tasks. When the SCB receives a query keyword from an external party, it constructs a search token so that the peer can search over the public ledger and then returns related encrypted pointer. The SCB can further decrypt the pointer for the external party.
- **Elastic off-chain storage:** The off-chain cloud-based storage engine stores the encrypted data under an ABE scheme and responds to the SCB's (and external parties with granted rights) requests and queries.
- **Data user:** If an external party wants to search the data that contains a particular keyword, he/she has to submit this query keyword to the SCB. After searching, the SCB returns the encrypted data that contains this keyword to the user. As for internal private channel users, e.g., a device, it can directly request the private ledger peer to return the attestation result and plaintext pointer.



SCB

ASSURED DATA STORAGE ENGINE

ASSURED PUBLIC LEDGER

EXTERNAL PARTIES

ASSURED PEERS

Send Attestation Bundle
(1) Attestation Results
(2) Attestation ID
(3) ABE encrypted Attestation Raw Data
(4) Meta data

Send Encrypted Attestation Raw Data and Request Pointer

Store the Encryption and Generate Pointer

Return Pointer

(1) Encrypt the Pointer
(2) Construct Keyword dictionary
(3) Construct DSSE Bundle

Send the DSSE Bundle: (Encrypted Pointer and Encrypted Index Structure) And Keyword Dictionary

Peer(s) Merge the Bundle and Dictionary into the Public Ledger

DSSE CONSTRUCTION AND STORAGE

Check Keyword Dictionary

Request Search Token based on Keyword(s)

Construct Search Token

Send Search Token

Peer(s) Search Matching Encrypted Pointers based on the Token

Return Encrypted Pointer

Decrypt the Pointer

Send the Pointer

Request Data via the Pointer

Return Encrypted Attestation Raw Data

Decrypt the Data with help of TPM

DSSE DATA RETRIEVAL

Figure 17. General Workflow of DSSE Secure Data Storage and Search on Public Ledger

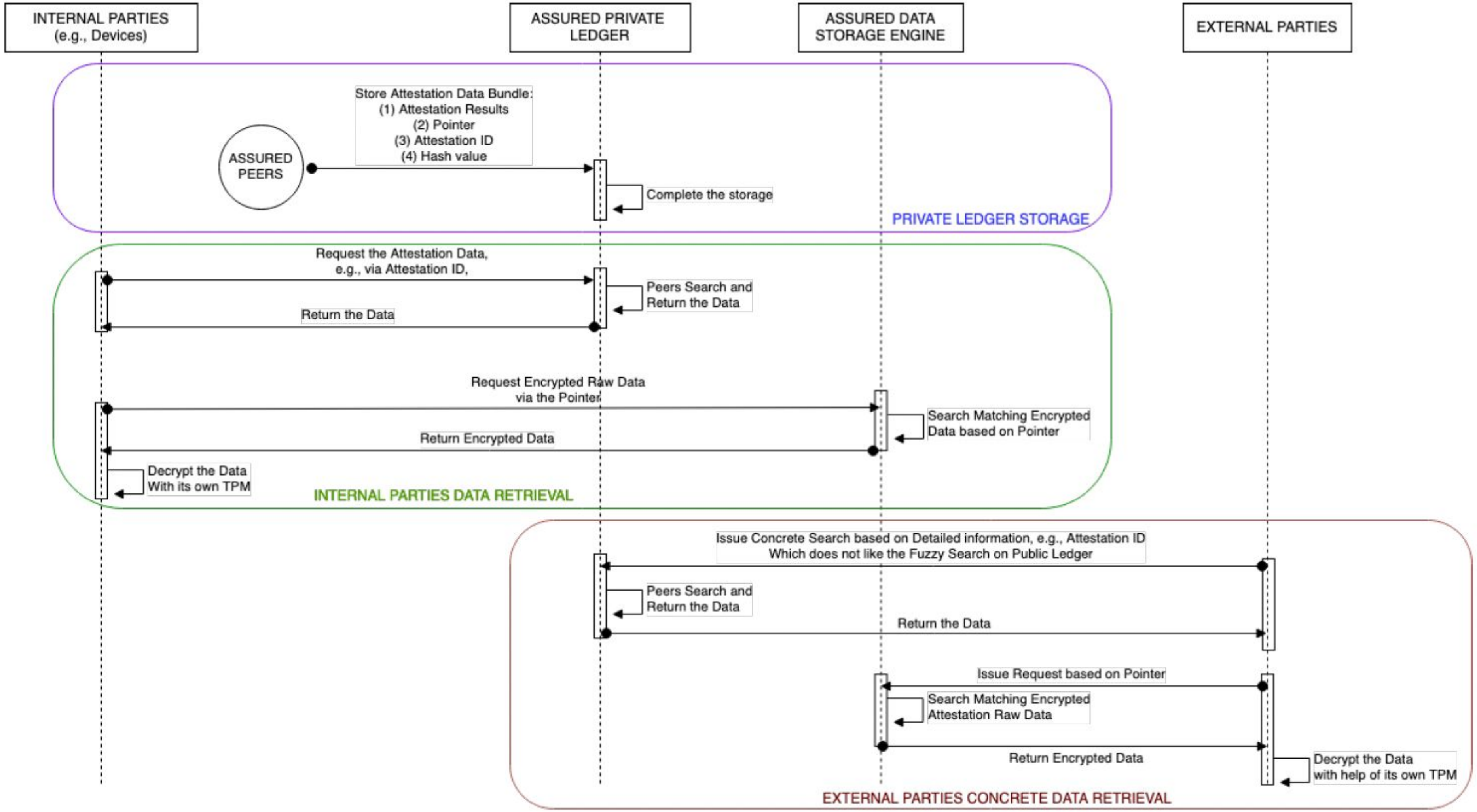

INTERNAL PARTIES (e.g., Devices)
ASSURED PRIVATE LEDGER
ASSURED DATA STORAGE ENGINE
EXTERNAL PARTIES
ASSURED PEERS
Store Attestation Data Bundle:
(1) Attestation Results
(2) Pointer
(3) Attestation ID
(4) Hash value
Complete the storage
PRIVATE LEDGER STORAGE
Request the Attestation Data, e.g., via Attestation ID,
Peers Search and Return the Data
Return the Data
Request Encrypted Raw Data via the Pointer
Search Matching Encrypted Data based on Pointer
Return Encrypted Data
Decrypt the Data With its own TPM
INTERNAL PARTIES DATA RETRIEVAL
Issue Concrete Search based on Detailed information, e.g., Attestation ID Which does not like the Fuzzy Search on Public Ledger
Peers Search and Return the Data
Return the Data
Issue Request based on Pointer
Search Matching Encrypted Attestation Raw Data
Return Encrypted Data
Decrypt the Data with help of its own TPM
EXTERNAL PARTIES CONCRETE DATA RETRIEVAL

Figure 18. General Workflow of Secure Data Storage and Search over the Private Ledger

PARTNERS

UBITECH ubiquitous solutions

MARTEL innovate

Suite5 We Deliver Intelligence

netcompany intrasoft

SPACE

United Technologies Research Center

uni.systems

BIBA Bremer Institut für Produktion und Logistik

CITY OF ATHENS ΔΑΕΜ CITY OF ATHENS - IT COMPANY

TECHNISCHE UNIVERSITÄT DARMSTADT

TU Delft

DTU

UNIVERSITY OF SURREY

NVIDIA

ASSURED

Q&A

PROJECT-ASSURED.EU

@Project_Assured

ASSURED project is funded by the EU's Horizon2020
programme under Grant Agreement number 952697

THANKS

PROJECT-ASSURED.EU

@Project_Assured

ASSURED project is funded by the EU's Horizon2020
programme under Grant Agreement number 952697