This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement Num. 871754 Ref. 2020-2022 - 31/12/2022

i3-MARKET

Intelligent, Interoperable, Integrative and deployable open source MARKETplace

Deliverable D4.2 – Decentralised Storage Specification and Reference Implementation

Deliverable No.	D4.2	**Due Date**	31-Dec-2022
Description	Decentralised Storage Specification and Reference Implementation		
Type	Other	**Dissemination Level**	Public
Work Package No.	WP4	**Work Package Title**	i3-MARKET Backplane for the Integration of Data Platforms and Market Economy
Version	Dec 2022	**Status**	Final

Authors

Name	Partner	e-mail
Kaarel Hanson	GUARDTIME	Kaarel.hanson@guardtime.com
Andres Ojamaa	GUARDTIME	andres.ojamaa@guardtime.com
Vladimir Rogojin	GUARDTIME	Vladimir.rogojin@guardtime.com
Jonathan Hepp	GUARDTIME	jonathan.hepp@guardtime.com
Juan Hernandez Serrano	UPC	j.hernandez@upc.edu
Rafael Genés Durán	UPC	rafael.genes@upc.edu
Jose Luis Muñoz Tapia	UPC	jose.luis.munoz@upc.edu

History

Rev.	Author(s)	Organisation(s)	Date	Comments
V0.1	All	GUARDTIME, UPC	25.06.2021	First document Release
V0.2	Andres Ojamaa	GUARDTIME	27.09.2021	Revision of the first Release
V0.3	Jonathan Hepp	GUARDTIME	15.05.2022	Verifiable Data Integrity solution description
V0.4	Kaarel Hanson	GUARDTIME	28.09.2022	Document alignment with applied changes to the implementation
V0.5	Andres Ojamaa	GUARDTIME	05.12.2022	Revision of the final Release

Key data

Keywords	Data Storage, blockchain, decentralised storage, distributed storage
Lead Editor	Name: Kaarel Hanson Partner: Guardtime
Internal Reviewer(s)	Juan Hernandez Serrano, UPC

Glossary

AB	Advisory Board
AMR	Annual Public Management Report
DESCA	Development of a Simplified Consortium Agreement
DoW	Description of Work
EC	European Commission
EU	European Union
EUPL	European Union Public Licence
FIA	Future Internet Assembly
GA	Grant Agreement
GNU	Generic Public Licence
GCC	GNU Compiler Collection
IERC	European Research Cluster on the Internet of Things
IMR	Interim Management Report
IP	Internet Protocol
IPR	Intellectual Property Rights
ITU-T	International Telecommunications Union
OGC	Open Geospacial Consortium
PMR	Periodic Management Report
PC	Project Co-ordinator
PMO	Project Management Office
PM	Project Manager
PO	Project Officer
PHP	Hyper Text Procesor
QA	Quality Assessors
SB	Supervisory Board
SSN	Semantic Sensor Networks
STREP	Specific Targeted Research Project
TL	Task Leader
TMB	Technical Management Board
WP	Work Package
WPL	Work Package Leader
W3C	World Wide Web Consortium

Statement of originality

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

Abstract

Every federated information system requires means to store and share data securely. i3-MARKET network is not an exception; hence, a well-thought solution that is secure, reliable and usable by all entities in the i3-MARKET network, is needed. The aim of data storage is to store common data in a federated network of data marketplaces. The common data shared between participating data marketplace instances may include identity information, shared semantic models, meta-information about data sets and offerings, semantic queries, sample data, smart contract templates and instances, crypto tokens and payments. No single party should fully control the data storage system and there shall be no single point of failure. In order to fulfil the needs of the aforementioned data types, two separate storage solutions are used: the decentralised and the distributed one.

The former supports the management of distributed identities and smart contracts. However, the latter has an important role in data synchronization between different i3-MARKET nodes, and, optionally, storage of data sets on sale. Moreover, the distributed storage supports non-repudiation service and auditable accounting features of i3-MARKET.

The design of the distributed storage has been an iterative process. As the key component using the storage is the Semantic Engine, data synchronisation has been the major topic of discussion between Tasks 4.1 and 4.2. Several solutions were analysed and proposed, resulting in the implementation of a federated query engine index.

Data storage system takes full advantage of available base technologies and builds on top of these in order to satisfy i3-MARKET needs and requirements, with a focus on federated system architecture. The underlying technologies chosen for decentralised and distributed storage means are Hyperledger BESU and CockroachDB, respectively.

Design and analysis work have taken majority of the time, due to trade-offs made in order to accommodate the needs of all relying parties.

The federated query engine index management solution is available and integrated into the i3-MARKET network, deployed as a smart contract on Hyperledge BESU. Moreover, a solution called Verifiable Data Integrity has been implemented on top of Auditable Accounting to further increase the reliability of data. And finally, access management solution governing the data access has been designed and implemented, relying on reliable and secure key management solution.

Table of contents

List of tables

List of figures

1 About This Document

This is the final version of specification of Task 4.1, Decentralised data storage specification FINAL.

1.1 Deliverable context

Project item	Relationship
Objectives	TO.6 – Decentralised and distributed storage solutions provided by Task 4.1 are the base technologies supporting the implementation and operation of a trusted, interoperable and decentralised infrastructure.
Exploitable results	Marketplace Backplane The deliverable contributes to marketplace backplane by providing the underlying storage mechanisms for sharing data between marketplaces, managing identities, storing SLSs, verifiable claims, etc.
Work plan	The current deliverable is an output of T4.1 (WP4).
Milestones	The current deliverable is linked to milestone MS10.
Deliverables	The deliverable has direct relationships with the following deliverables: - D2.2 - D2.3 - D3.1 - D3.3 - D3.5 - D4.3 - D4.4
Risks	Current deliverable mitigates the following risks from the list of Critical Implementation risks: 1, 2, 5, 8, 11, 14, 15

Table 1. Deliverable Context

1.2 Introduction

The current document presents the specification of the data storage component of i3-MARKET.

Section 2 provides the objective, architectural view and security aspects. Moreover, objectives are broken down into separate releases. The main technical contribution to the decentralised and distributed storage state-of-the-art is also outlined in Section 2.

Section 3 describes the features and use cases that are implemented and supported by the component.

Section 4 presents the sequence diagrams of the features provided by the component, to give a better overview of the interactions between the different components that the storage is connected to in i3-MARKET.

In Section 5, a brief overview of the interfaces of implemented solutions is given.

Section 6 gives a brief overview of the technologies used for the implementation of data storage component.

In Section 7, the technical contributions of i3-MARKET project, in the context of decentralised and distributed storage, is provided.

Section 8 concludes the work done.

2 System Specification

2.1Objectives

The objective of Task 4.1 is to provide the data storage system for the i3-MARKET framework to store common data in a federated network of data marketplaces. The common data shared between participating data marketplace instances may include identity information, shared semantic models, meta-information about data sets and offerings, semantic queries, sample data, smart contract templates and instances, crypto tokens and payments. No single party should fully control the data storage system and there shall be no single point of failure.

The high-level capabilities that the data storage aims to provide are:

1. Decentralised Storage
2. Distributed Storage

The Decentralised storage shall provide highest available security guarantees in a federated network. The Decentralised storage subsystem is built on a secure Byzantine fault tolerant consensus based distributed ledger. Due to high security requirements, the performance and storage space of such a system may be relatively limited compared to conventional databases.

The Distributed storage shall provide a database-like subsystem that is scalable, deployed on all i3-MARKET nodes, has a rich query interface (SQL) and can handle large amounts of data while the i3-MARKET shall rely on the API of the decentralised storage provided out-of-box.

2.1.1 Task Objectives R1

The objectives of T4.1 in R1 include both, decentralised and distributed storage. However, as the storage layer is dependent on the components that require storage features, the objectives are completed incrementally in coherence with the building blocks.

In the first release, the technologies for both capabilities were decided. The decision was made according to the technology needs of the relying building blocks of i3-MARKET. Initially, the aim was to deploy and configure the base technologies on the i3-MARKET network infrastructure for the first release, but the task was postponed to the second release due to unavailability of the infrastructure.

In terms of the features and functionality provided by the data storage, the decentralised storage supports the management of DID documents.

In R1, no specific implementation was done in the context of the data storage system, however, it was foreseen that the storage system implements functionality in subsequent versions. More specifically, it was foreseen to involve the data storage system in the process of data synchronization between the distributed storage and local triple stores of semantic engines.

Two alternative approaches for data synchronization have been proposed, depending on the overall system architecture:

- Data storage layer hosts a global triple store in addition to the local triple stores in each i3-MARKET instance.
- Data storage layer consists of a distributed SQL database that distributes updates to local triple stores deployed in each marketplace instance.

2.1.2 Task Objectives R2

During the validation of the solutions proposed for data synchronisation between different i3-MARKET instances, a decision was taken to design a more optimal solution. The initial proposed concept foresaw that the data stored in each Semantic Engine would be duplicated on the distributed storage. Such approach initially included data backup without additional effort, but in the end, it was decided that mirroring data is not needed. Instead, the distributed storage shall provide features to support federated data discovery queries.

For this purpose, the distributed storage stores an index that is consulted by the Semantic Engine (SEED) in case a federated query is produced. The index (SEED Index) consists of data categories mapped to the endpoint location addresses of the corresponding Semantic Engines. Among the aims of R2 is also to finalise the implementation of the index and the interfaces required for managing the index. Data storage subsystem exposes at minimum two endpoints for index management. One of the endpoints is used to update the index upon reception of new offerings by the SEED and the second one to query the index itself. The index is necessary for the creation of a federated query that allows data discovery in every marketplace connected to the i3-MARKET network.

Moreover, it is intended to provide data set storage features for data owners and providers, who do not have the means or capacity to store the data that is on sale on any of the marketplaces connected to the i3-MARKET network. In order to manage access to such data stored in the distributed storage, a suitable solution is needed. For this purpose, data storage shall interface with the smart contract manager, handling access and permissions, in order to avoid illegal or undesired access to the data.

2.1.3 Task Objectives R3

Once the validation of i3-MARKET nodes with the pilots began, it was soon discovered that a distributed database is not a secure solution in a federated data market network. If each database instance has the same privileges, one malicious actor in the network can destroy the entire distributed database. Furthermore, connecting a new node to the network is not an automated process, but requires configuration by an i3-MARKET system administrator. This is in conflict with the nature of i3-MARKET, which in essence should be decentralised system without any central authority or requiring any system maintenance by the initial system producers.

Therefore, a decision was made to disconnect each CockroachDB instance in i3-MARKET nodes. This, however, had impact on couple of services that had relied on the distribution of data among all nodes. First of all, federated query engine index had to be redesigned. A decision was made to migrate the index from distributed storage to decentralised storage. For this purpose, a smart contract was implemented that stores the URI of each of the SEED instance in the i3-MARKET network along with available semantic data categories belonging to the specific SEED instance. As the index is relatively small and mostly read operation is expected, no serious drawbacks to using a blockchain have been discovered.

Secondly, the Auditable Accounting was affected by the change, however, in collaboration with the partners, the solution was redesigned to work on an independent database without sacrificing any of the initial and planned functionality.

As an additional means to provide data security in terms of verifiable integrity, Verifiable Data Integrity (VDI) was implemented. The purpose of the VDI is to provide an infrastructure for data to be stored in a way that its presence, or lack thereof, can be cryptographically proven. It takes advantage of blockchain technology to determine the integrity of the data it contains.

2.2 Context

Figure 1 shows the data storage system in the i3-MARKET context. All components that need to persist some global state or use global data for operation will interact with the data storage system. The data storage system needs do interface with the Semantic engine system and the Trust, security and privacy system for access management.

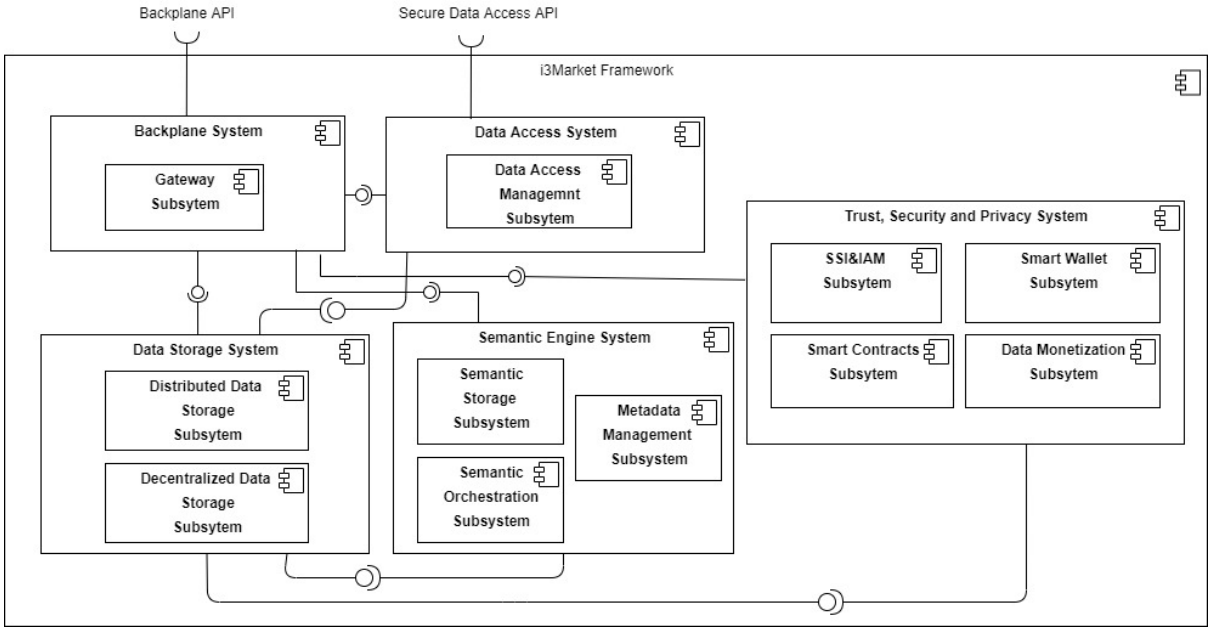

Figure 1. Data Storage System in i3-MARKET context

2.3 Building block big picture

The Storage system consists of two main subsystems for implementing the decentralised storage and distributed storage features, respectively. The subsystems are relatively independent of other systems and also with each other.

The diagram of Decentralised storage subsystem is shown in Figure 2. The Decentralised storage subsystem is implemented as a blockchain-based distributed ledger network. The software implementation is Hyperledger Besu in a permissioned setup using IBFT 2.0 consensus. Hyperledger Besu uses internally an embedded RocksDB instance for storing linked blocks (the journal of transaction) and world state (the ledger). Hyperledger Besu can instantiate and execute smart contracts for supporting the use cases of i3-MARKET framework.

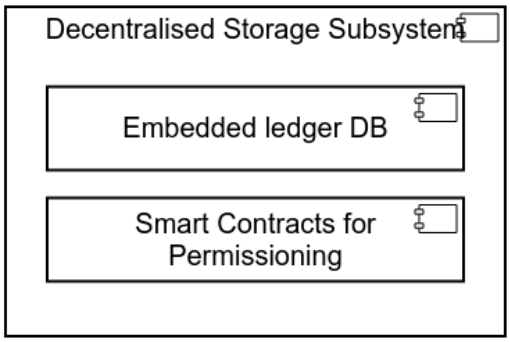

Figure 2. Decentralised Storage Subsystem

The components depending on the decentralised storage subsystem will use Hyperledger BESU's native JSON-RPC-based interface. A separate interface layer for accessing (or limiting access to) decentralised storage is not planned, as the nodes of the decentralised storage will already validate all transactions submitted to the ledger.

The diagram of Distributed storage subsystem is shown in Figure 3. The subsystem consists of database nodes. The database provides an SQL interface to other i3-MARKET framework components. The software implementation database is CockroachDB that can be accessed via PostgreSQL-compatible wire protocol for which a large number of client libraries exist in different languages and platforms. Only secure access (TLS with mutual authentication) to the database will be enabled, hence all clients need to use private keys and valid certificates to access the database.

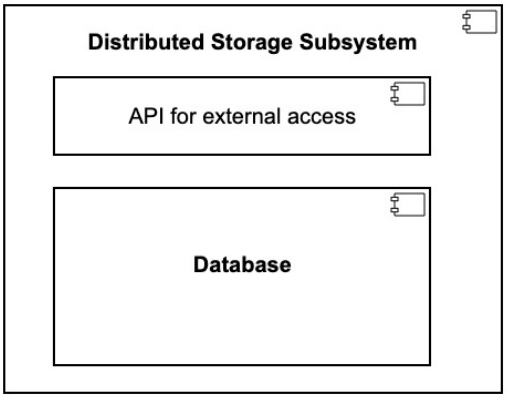

Figure 3. Distributed Storage Subsystem

2.4 Security

2.4.1 Authentication and Authorization

The distributed storage component is an internal component with no external access. That is to say that it will have connections only with other trusted services within the i3-MARKET backplane. Even though this simplifies the necessary measures in terms of authentication and authorization, it is still needed to secure machine-to-machine connections between the i3-MARKET services, since they can be deployed on shared infrastructure.

The authentication and authorization solution relies on providing the distributed storage behind a TLS server endpoint and requiring TLS client certificates for the different connecting services. The setup will guarantee end-to-end security between the distributed storage service and any of its client services.

The governance of the certificates has followed up to now the *keep it simple* approach. The Distributed Storage system will be in charge of issuing the servers' and clients' certificates, unless the instance has its own Certificate Authority (CA), in which case the CA is responsible for issuing server certificates to the distributed storage server component and client certificates to the clients.

2.4.2 Service Availability

The storage subsystem is a critical component of the i3-MARKET network contributing to the proper functioning of the platform. Hence, appropriate measures in the form of design, choice of technologies and deployment, have to be applied. Fortunately, the two main subsystems used in the storage solution already have strong built-in availability features that will be summarised below.

2.4.2.1 Distributed Storage

The distributed storage solution is based on a CockroachDB server. Initially, the database was deployed as a global cluster of database nodes, however, after the initial testing of the entire network, couple of issues was discovered. First, the deployment of a CockroachDB instance and connecting it to a cluster is not an automated process, but rather manual as configuration steps must be tightly coordinated between the nodes. This contradicts with the overall concept of i3-MARKET, which should be operable without any central administration.

The second and far greater problem, that was eventually acknowledged, is that each node in a CockroachDB cluster has equal rights with full administrative privileges over the cluster. This is a problem, because any node can alter data and there is no consensus mechanism to agree on the changes. Furthermore, in case of the rise of a rogue node could potentially lead to full erasure or silent corruption of the entire database.

Therefore, a decision was made to replace the global cluster with independent clusters deployed at each i3-MARKET instance. In this deployment mode, each instance is responsible for its own operation and a configuration mistake in one instance or a malicious act cannot affect the stored data at other instances. There was only one implication to this change – SEED Index would not work in such a setup anymore. As a result, the index was migrated from the Distributed Storage to the Decentralised storage.

2.4.2.2 Decentralised Storage

The decentralised storage used in the platform is a Hyperledger BESU network which uses the IBFT 2.0 (Proof of Authority) consensus protocol. In this network, there are 4 validator nodes based on the genesis configuration stored in the corporative Nexus. In this configuration, there are 3 accounts to be used by the i3-MARKET federation.

The federated search engine index service uses the Hyperledger BESU blockchain as its storage backend. For this purpose, a smart contract storing the endpoints of all SEED instances along with the associated data categories has been deployed on the blockchain.

In this scenario, different components like auditable accounting, SEED Index, etc. are capable to deploy and manage smart contracts and transactions over those accounts.

2.4.3 Verifiable Database Integrity

2.4.3.1 Abstract

The purpose of the VDI is to provide an infrastructure for data to be stored in a way that its presence, or lack thereof, can be cryptographically proven. It takes advantage of blockchain technology to determine the integrity of the data it contains.

The VDI component consists of a library that implements a Compact Sparse Merkle Tree (CSMT)[1] and exposes an API that allows for data to be inserted, retrieved, updated and removed. The API consumer can later obtain proofs of membership/non-membership and verify those proofs against the existing Merkle tree.

This data structure works on the same principles of Verifiable Maps[2], periodically generating a root hash that, after been made public, can guarantee the integrity of the data at that point in time. Membership or non-membership of any given key can be cryptographically proven against the Merkle tree.

2.4.3.2 Integration

The VDI is not a separate application on its own but is integrated into the Auditable Accounting component, as can be seen on Figure 4. The MerkleTreeService class exposes methods to create a CSMT tree from the array of hashes obtained from registries. The individual proofs are then stored along with their corresponding root hash into the registries repository in the database. The unregistered blockchain record is also stored in the database along with the serialised Merkle tree data itself.

[1] Compact Sparse Merkle Trees: https://eprint.iacr.org/2018/955.pdf

[2] Verifiable Data Structures, pg. 2: https://continusec.com/static/VerifiableDataStructures.pdf

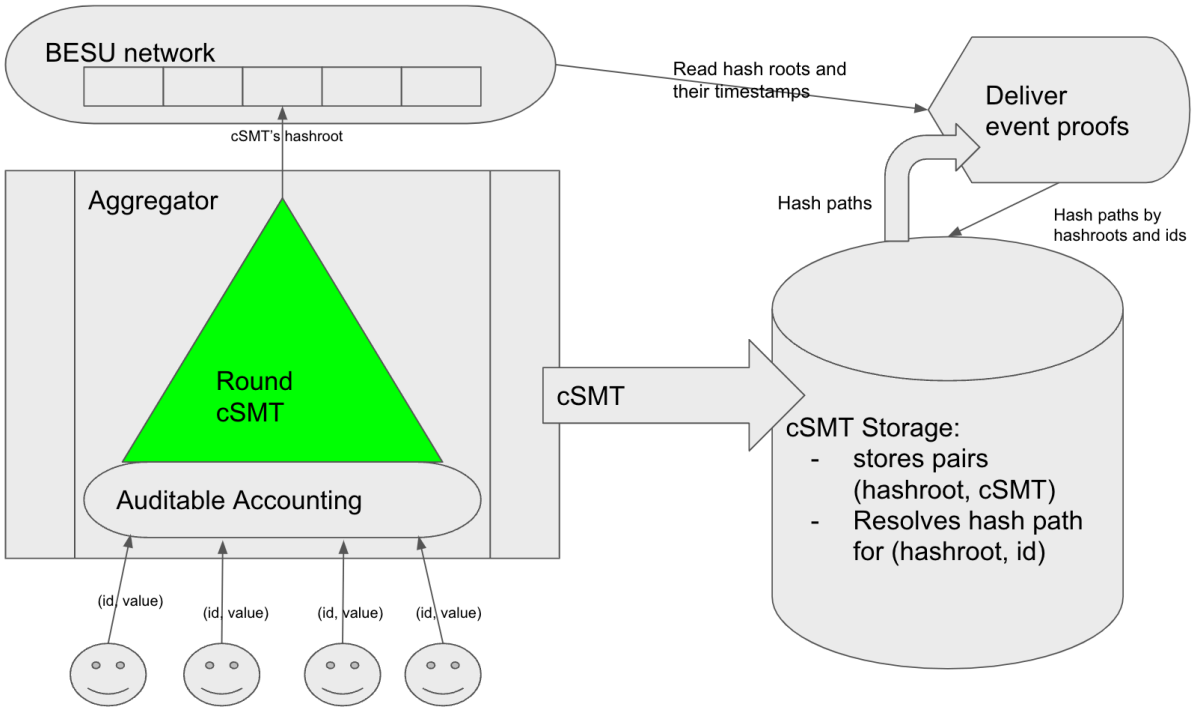

Figure 4. VDI Integration with Auditable Accounting

3 Use Cases / Features

The decentralised storage sub-component of data storage provides functionality to manage an index used for semantic data discovery. The federated query engine index supports federated queries, a concept implemented by the Semantic Engine. The distributed storage plays a vital role in supporting the verifiable data integrity, non-repudiation service and auditable accounting.

3.1 Federated Query Engine Index management

Decentralised storage implements two main use cases – managing the index and querying the index – in order to provide the required functionality to the Semantic Engine for accessing the content of the index.

The index is a collection of data categories together with the endpoint location addresses of the corresponding Semantic Engines. One Semantic Engine is not limited to storing offerings belonging to one category, but to several of them. Hence, the index contains one to many relationships, linking a specific Semantic Engine to a set of data categories.

Each Semantic Engine instance has a private key, while the corresponding public key serves as an identifier that is associated with a set of SEED Index records. The private key is needed to update corresponding index records. Moreover, in order to pay for update transaction, the SEED account must have enough resources. The owner of the SeedsIndexStorage smart contract can assign administrator roles to other keys that can update records stored under any public key.

Every new marketplace joining the i3-MARKET network, will connect to the decentralised storage through Semantic Engine. If the marketplace has been around for a while, the marketplace has most probably stored offerings metadata. This metadata should also be stored in the SEED to participate in federated queries. Therefore, such marketplace would have to populate the index by inserting category information to the decentralised storage.

3.1.1 Manage

In order to provide the most recent and accurate information to the Semantic Engines in the i3-MARKET network, the index must be kept up to date at all times. Therefore, functions – insert, update and delete - to maintain the index are required. All these activities are limited to registered i3-MARKET nodes only and the authentication uses self-signed certificates.

3.1.1.1 Insert

Before an i3-MARKET instance receives any data offering registrations, the Semantic Engine has no reason to insert any content into the index. Although it is possible to insert an empty entry containing the endpoint address and an empty category list to the index, it is recommended to keep the index clean of unnecessary information. After receiving the first data offering, the Semantic Engine inserts the first entry to the index, revealing to other i3-MARKET instances the category of offerings stored in that specific Semantic Engine.

3.1.1.2 Update

Over the course of the market lifecycle, data offerings of different data categories are stored in a single marketplace. Upon the registration of a data offering belonging to category that is not yet present in the Semantic Engine, the Semantic Engine updates the index with relevant information (data category, endpoint address, etc.) by inserting a new entry to the database.

3.1.1.3 Delete

The final management activity of the index lifecycle allows the removal of entries from the index. It is the responsibility of the Semantic Engine to keep the index up to date, therefore, redundant and outdated information is removed from the index. In the event of closing down of an i3-MARKET marketplace instance, either temporarily for maintenance or indefinitely, the Semantic Engine has to remove unavailable content from the index. Moreover, this function should be accessible by a system administrator to remove relevant entries from the index, in case of a sudden shut down of a marketplace/i3-MARKET node.

3.1.2 Query

In case a Semantic Engine needs to perform a federated query among all other instances in the i3-MARKET network, the index shall provide input to the federated query. The Semantic Engine firstly queries the index with relevant parameters (data category, description, etc.) and the distributed storage shall return information from the index indicating which i3-MARKET instance contains the data that the SEED is looking for.

3.2 Verifiable Database Integrity

The library[3] is implemented in Typescript and distributed as a node package. The main data structure of the VDI is a class called CSMT. This class keeps a map of all the nodes that are stored in the tree as well as the tree's root hash. On an empty tree, the root hash is initialized as a zero node. Data (in key-value format) can then be inserted into the tree, producing new nodes and updating the root hash.

What follows is a summary of the functions that are available for the consumers of the CSMT class.

3.2.1 Add

Adds a given key-value pair to the tree. The key has to be in byte array format. The value can be any arbitrary string and is optional. If the provided key already exists in the tree, an error is returned, no duplicate keys are allowed. The data is combined in a new array that contains the key in hexadecimal format, the hash of the value and an entry mark that flags this as a leaf node. This data is then inserted into the nodes map, where the hash of the data is, in itself, the key for this record in the map. Finally, the tree's root hash is recalculated.

3.2.2 Insert

This is a convenient method to insert data in bulk to the tree. This method just validates the data and calls the method **add** above on each individual element.

3.2.3 Get

Looks up for a given key in the nodes map. Returns the hash of the corresponding value for the key, or **undefined** if the key is not present.

[3] https://github.com/i3-Market-V2-Public-Repository/SP4-VerifiableDatabaseIntegrity

3.2.4 Delete

Looks up for a given key in the nodes map and removes it. The tree's root hash is then recalculated based on the remaining nodes. If no key is found an error is returned.

3.2.5 Create Proof

Looks up for a given key in the nodes map and creates a new proof object. The proof object contains the data itself (if present), the chain of additional nodes along the tree traversal, the root hash of the tree, and a membership flag (true if the given key is present in the tree, false otherwise).

3.2.6 Verify Proof

When the consumer has a proof object, it can verify if that proof matches against the existing tree by calling this method. It verifies if the root hashes and node chain (in case of membership) matches with what the CSMT class has stored internally. It returns true if the proof matches. If it returns false, it means that either the proof does not belong to this tree or that the proof was tampered with.

3.3 i3-MARKET backplane use cases

It is expected that several services of the i3-MARKET Backplane use the Distributed Storage for reliably storing metadata regarding the provided service. The following services have direct or indirect access to the distributed storage: the non-repudiation service and the Auditable Accounting service.

3.3.1 Non-repudiation service

The distributed storage will be used by the Non-Repudiation protocol for reliably storing proofs of the data exchanges between i3-MARKET consumers and providers. The non-repudiation protocol has been developed in T3.2, and it is presented in deliverable D3.4 "Smart contracts and wallets specification and reference implementation report V1". The proofs that make possible a reliable billing of the service is expected to be stored by consumers and providers in their wallets and a secure backup in the distributed storage is currently under development. The provision of either direct access to the distributed storage or indirect access through the Auditable Accounting system is still to be decided, but in any case, only the involved stakeholders in every data exchange should gain access to the non-repudiation proofs.

3.3.2 Auditable accounting

The Auditable Accounting component is responsible for logging relevant information that occurs in the ecosystem for data marketplaces, and in this way, it enhances the trust in the platform. Our solution must ensure the enforcement of the Data Sharing Agreement (DSA) terms, agreed upon by all involved parties, by recording them in an auditable, transparent, and immutable way. Smart Contracts are the key part of the proposed solution for auditable accounting. The auditable accounting component is an abstraction layer to access the Smart Contracts and to allow the integration with the rest of the platform. The auditable accounting component is a service which includes an API to automate the process of logging and auditing interactions between components and record the registries in the Blockchain.

To allow external parties to check that logs have been properly registered in the blockchain, interested parties need to obtain certain data from the distributed ledger as well as some off-chain data provided by the Auditable Accounting module via an API. This off-chain data are

essentially Merkle proofs for each individual record (more details about this module including its implementation and architecture are provided in the deliverable D3.3).

In this context, it is important that the off-chain data are provided with high availability. For this reason, the Auditable Accounting module uses the distributed storage component. In this way, high availability and data replication is provided to the relevant off-chain data required to store the registries and verify auditable logs.

4 Sequence Diagrams

4.1 Federated Query Engine Index Management

The sequence diagram in Figure 5 shows the interaction of the decentralised storage on the SEED regarding the federated query engine index management. Each function – insert, update, delete and query – has been depicted on a single sequence diagram, as there is no relevant complexity to be shown for each interaction. Index record identifiers (*uuid* in the figure) are derived from node public keys via cryptographic hashing and all requests must be signed with an authorized key (e.g. corresponding private key).

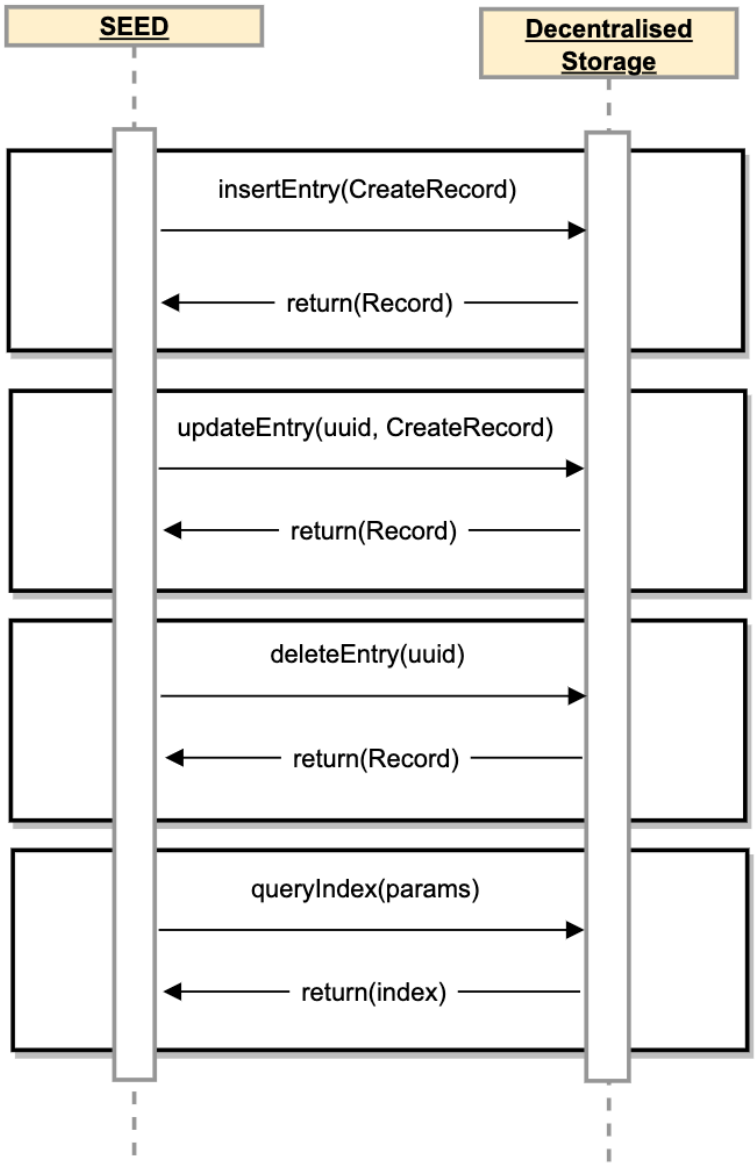

Figure 5. Federated Query Engine Index Management

4.2 Verifiable Data Integrity

The sequence diagram on Figure 6 demonstrates the integration of Verifiable Data Integrity with the Auditable Accounting sub-system. All features are displayed on a single diagram, as there is not specific complexity within the functions.

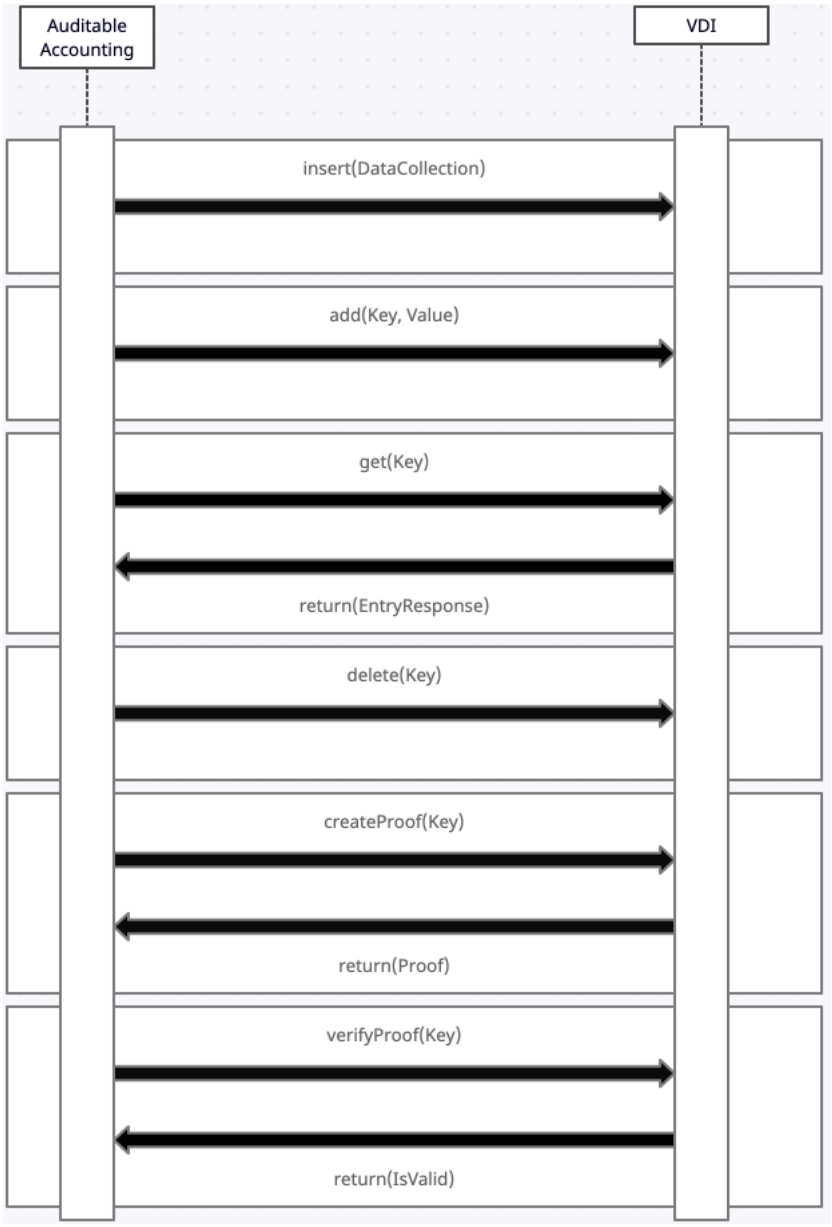

Figure 6. Verifiable Data Integrity

5 Interfaces Description

The distributed storage sub-system does not expose a bespoke API for internal or external services. Each system within the backplane uses the storage component's out-of-box means for connectivity.

Likewise, the API provided by the decentralised storage comes out-of-box with the solution. The service can be accessed via JSON-RPC protocol offered by Hyperledger BESU. Please refer to *BESU Documentation*[4] for the details of the API provided by Hyperledger BESU and client libraries.

The SEEDS Index solution consists of a Java library, called SeedsIndex, that provides wrappers for the smart contract and utility functions for convenience. The SeedsIndex library uses Web3j[5] library for accessing the BESU network. The library interface is documented by extensive Javadoc comments and a complete usage example included in the library.

4 Hyperledger BESU documentation, https://besu.hyperledger.org/en/stable/

5 https://www.web3labs.com/web3j-sdk

6 Selected Technologies

The partners within Task 4.1 have taken the following decisions for the choice of selected technologies in order to satisfy the high-level capabilities of T4.1:

- Hyperledger BESU to satisfy decentralised storage
- CockroachDB[6] to satisfy storage requirements

Hyperledger BESU is an open-source Ethereum client. The decision to select Hyperledger BESU to satisfy the needs for decentralised storage has been made based on the following assumptions:

- Self-sovereign identity and access management (T3.1) has decided to base the reference implementation on Veramo, which specifically requires Ethereum-based blockchains.
- Auditable accounting and data monetisation require smart contract whose functionalities are easily satisfied on Ethereum-based blockchains.

CockroachDB is a relational database management system. CockroachDB has been chosen as the storage solution due to previous experience of the technology by partners in T4.1. Moreover, it is highly scalable, designed to deliver fast access and resilient to network outages. The only shortcoming of the chosen technology is the lack of features guaranteeing data integrity in case of the presence of malicious (e.g., because of honest mistakes or sophisticated external attacks) users. As all nodes of a CockroachDB cluster that is a part of an i3-MARKET instance are under the control of that instance, this shortcoming is not relevant in the i3-MARKET architecture. For client authentication in CockroachDB, mutual TLS authentication is used.

The semantic search engine index is implemented as a smart contract (written in Solidity programming language), which is deployed on BESU blockchain for distributed access. Updates to the index are authorized with digital signatures.

6 CockroachDB, https://www.cockroachlabs.com/product/

7 Testing Strategy

Data storage components mainly comprise of commercial off-the-shelf software, therefore, CockroachDB and Hyperledger BESU are not tested explicitly, but indirectly with the testing of applications and components that rely on the storage.

SEED Index and Verifiable Data Integrity have been implemented within the project, which required at least some level of validation. For validation of the implementations, unit testing with additional manual and scripted integration testing was used. Once the components were deployed and integrated, they started getting continuous use by other components of the system. This helps to ensure that the software components are in a good working condition and any changes and updates will also get tested by the integration tests of other components which means regressions are likely to be discovered in the testing environment.

8 Technical contributions of i3-MARKET project

Data Storage subsystem has a vital role in i3-MARKET network, providing both structured database and ledger storage solutions, hence supporting several use cases. There are several contributions that the Data Storage subsystem brings to the i3-MARKET project.

The main technical contribution of i3-MARKET project for decentralised and distributed storage is combining existing state-of-the-art storage technologies in a novel way and enriching these components with custom application-specific logic and interfaces to support the diverse goals of the i3-MARKET framework that an off-the-shelf solution would not be able to cover. This can demonstrate how technologies with different technical characteristics can be used synergistically in a distributed system to leverage the relative strengths of each storage component in order to achieve new qualities in the whole system.

In particular, the decentralised storage component Hyperledger BESU already provides storage with strong security guarantees (integrity, availability), but due to its architecture, it is relatively inefficient and expensive for storing large or mutable data sets. Also, the query processing capabilities of Hyperledger BESU are quite limited compared to a relational database engine. As the i3-MARKET framework contains elements of a conventional information system where such features are desirable, it is reasonable to integrate a database management system that provides efficient storage for large and mutable data sets as well as rich query processing facilities. An SQL database management system comes with mature software development tools and libraries while also being already familiar to software developers. However, even the state-of-the-art SQL database management systems, such as CockroachDB, that support redundancy and distributed deployments, lack some of the integrity guarantees of blockchain-based distributed ledgers. The integrity of these databases relies on the node operators being trusted. In case of an i3-MARKET like setting where participants may have conflicting financial interests and there is a possibility of collusion to achieve gains, it is important to minimise the amount of trusted components (“trusted component” means that users must rely on the component without being able to actually verify the correct operation of the component). By combining these technologies, the whole system can exploit a storage system with large capacity, powerful query processing and good integrity properties without relying on trusted components.

The auditable accounting component is a good example how the storage system can be utilized – small pieces of data (hashes) are anchored in the distributed ledger and larger proofs are stored in CockroachDB. Also, the storage supports the coordination between different instances of semantic search engines where all parties can verify that the database cluster nodes operated correctly and, for example, did not hide some records from some parties.

9 Conclusion

The architectural view of the data storage system has been defined and the core technologies to support the features of decentralised and distributed storage have been chosen and are Hyperledger BESU and CockroachDB, respectively.

Considerable effort was put on the solution to support data synchronization between the marketplaces connected to the i3-MARKET network. Several options were presented, however, in the end the decision landed on federated query engine index management solution. The decentralised storage maintains an index that is consulted by each i3-MARKET node in order to build federated queries across all marketplaces. The index is published in a smart contract deployed on Hyperledger BESU. The functionality of the federated query index can be expanded based on emerging needs and innovative features built into the i3-MARKET platform.

Moreover, in order to add another layer of security, in terms of integrity verification, a solution called Verifiable Data Integrity has been designed, implemented and integrated into Auditable Accounting. The solution allows cryptographic proof to verify the presence or lack of data in an infrastructure.

The authentication and security of machine-to-machine connections has been designed and is based on public key infrastructure (PKI). As i3-MARKET architecture must support a decentralized network, each i3-MARKET instance can set up their own PKI or use an existing certificate authority and publish their trust anchor (root certificate of a certificate authority) in decentralized storage so that other instances can verify certificate chains presented by all parties.