TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Marten Kask 183000IABM

BLOCKCHAIN-BASED MEMBERS MANAGEMENT FOR THE UNIFIED EXCHANGE PLATFORM

Master's Thesis

Supervisor: Ahto Buldas
PhD
Co-Supervisor: Margus Freudenthal
PhD

Tallinn 2020

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Marten Kask 183000IABM

UNIFIED EXCHANGE PLATFORMI PLOKIAHELAL PÕHINEV LIIKMEHALDUS

magistritöö

Juhendaja: Ahto Buldas
PhD
Kaasjuhendaja: Margus Freudenthal
PhD

Tallinn 2020

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Marten Kask

17.05.2020

Abstract

The goal of this thesis is to improve the existing model of UXP (Unified eXchange Platform) Registry to reduce the need to trust to the correct behavior of the Governing Authority whilst considering the advantages of new technologies. The thesis describes the blockchain-based solution for the UXP Registry that uses Hyperledger Fabric platform.

UXP Registry is a server that is used to manage security policy of the UXP instance and distribute global configuration to the security servers of all members. It acts as a trusted third party: all the members have to trust that it does not abuse its obligations. One way to reduce the risk that the operation of the UXP instance is interfered because of the centralized management, is to replace UXP Registry as trusted third party with a distributed ledger, e.g., blockchain.

Hyperledger Fabric is selected as a platform to develop blockchain-based UXP Registry. It is described how the smart contracts and other features of the platform can be used to deploy the current and perspective UXP Registry processes. It is discussed which more perspective improvements could be implemented when replacing centralized UXP Registry with a blockchain-based platform.

It can be concluded that the need to trust the correct behavior of the Governing Authority can be reduced if the blockchain-based UXP Registry is implemented. Mostly because blockchain helps to increase transparency in the actions accomplished through UXP Registry.

The thesis is in English and contains 50 pages of text, 5 chapters, 6 figures, 1 tables.

Annotatsioon

Unified eXchange Platformi plokiahelal põhinev liikmehaldus

Antud magistritöö eesmärk on täiustada olemasolevat UXP (Unified eXchange Platform) Registry mudelit, et vähendada usaldust juhtiva asutuse korrektse käitumise vastu, võttes arvesse tehnoloogia arengut. Töö kirjeldab plokiahelal põhinevat UXP Registryt, mis kasutab Hyperledger Fabric platvormi.

UXP Registry on server, mida juhtivad asutused kasutavad UXP instantsi turvapoliitika haldamiseks ning globaalse konfiguratsiooni jagamiseks kõikide liikmete turvaserveritele. See käitub kui usaldatav kolmas osapool, mis tähendab, et kõik liikmed peavad usaldama, et juhtiv asutus ei kuritarvitaks oma kohustusi. Üks võimalus, kuidas vähendada riski, et UXP instantsi töö ei oleks häiritud keskse juhtimise tõttu, on asendada UXP Registry kui usaldatav kolmas pool hajusraamatuga.

Üks hajusraamatu tehnoloogiatest on plokiahel - andmestruktuur, mis koosneb järjestikku andmeblokkidest, mis luuakse määratud aja või sündmuse tagajärel. Selles töös tutvustatakse plokiahela põhikontseptsioone ning turvaomadusi, samuti analüüsitakse mitmeid varasemaid plokiahelal põhinevaid lahendusi, mis on registritega seotud. Selleks, et leida sobivaim plokiahela lahendus, analüüsitakse ka olemasolevaid UXP Registry protsesse.

Hyperledger Fabric on valitud plokiahelal põhineva UXP Registry välja töötamiseks. Kirjeldatakse, kuidas nutilepinguid ja teisi selle platvormi erisusi saab kasutada, et juurutada plokiahelal põhinevad olemasolevaid ja perspektiivseid UXP Registry protsesse. Kirjeldatakse Hyperledger Fabricu komponente ning selgitatakse, kuidas neid kasutatakse välja pakutud UXP Registry arhitektuuris. Arutletakse, milliseid perspektiivseid täiustusi saaks veel teha, kui asendada keskne UXP Registry plokiahelal põhineva lahendusega.

Järeldatakse, et usaldust juhtiva asutuse korrektse käitumise vastu saab vähendada, kui juurutada plokiahelal põhinev UXP Registry. Eelkõige selle tõttu, et plokiahel aitab UXP Registry kaudu tehtavate toimingute läbipaistvust suurendada.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 50 leheküljel, 5 peatükki, 6 joonist, 1 tabelit.

List of abbreviations and terms

API	Application Programming Interface
DDoS	Distributed Denial-of-Service
DSES	Decentralized Service Eco-System
FaaS	Federation-as-a-Service
GDPR	General Data Protection Regulation
IoT	Internet-of-Things
NIIS	Nordic Institute of Interoperability Solutions
PoW	Proof of Work
REST	Representational state transfer
SLA	Service Level Agreement
SOAP	Simple Object Access Protocol
SWIM	System Wide Information Management
TLS	Transport Layer Security
UXP	Unified eXchange Platform
WSDL	Web Services Description Language

Table of Contents

List of Figures

List of Tables

1 Introduction

This chapter introduces the concept of this thesis. Research problem is outlined and research questions formulated, along with, the aim of the research and expected outcomes are defined with the presentation of the proposed research methodology.

1.1 Background of the X-Road and the UXP

The X-Road project was commenced in Estonia to enable secure data transfer between different state databases in 2001[2, pp. 572–573]. Estonia had multiple state registers digitized, but these systems did not have efficient interoperability with each other. Legal and technical reasons were the main aspects why the registers were not effectively integrated. The biggest challenge in the technical view were that the registers were developed independently, i.e., different technologies were used. Hence, there existed a need for a platform which would allow to interconnect different information systems regardless their technical architecture.

Moreover, state registries often process personal data that might be needed in real time to make significant decisions, thus, high security requirements had to be taken into account during the development of the X-Road [2, pp. 572–573]. The initiation of X-Road also made possible to share obligation of offering public services with private sector, considering that public sector does not have enough capability to serve the purpose solely [3, p. 36]. To accomplish the secure technology for data transfers between registries, distributed architecture is used in the design of X-Road [4, p. 567]. The pattern where databases are centralized had to be prevented, as the whole X-Road platform might be affected in case of failure in the central database. High availability and scalability are another essential properties of X-Road: multiple servers can be used in parallel to distribute the load between servers and increase the data processing capacity.

The main components of the X-Road are security servers that are associated with various databases [4, p. 567]. The information systems can exchange data through the security servers which communicate directly with each other using cryptographically protected channels [5, p. 2795]. The functionality of X-Road is grounded on the service-based principle. Hence, services are provided over common standards, e.g., SOAP (Simple

Object Access Protocol) and WSDL (Web Services Description Language), to enable data processing operations, i.e., writing data, reading data and developing other data based business logic. Security servers are fundamental to allow integration of databases as the particular servers encrypt and decrypt communication, keep the records of exchanged messages and authorize organizations access to the data. Registry of certified security servers is managed and distributed to other security servers by the central server. For monitoring status of security servers, monitoring servers are used.

The X-Road has been in continuous development. The first version supported exchanging messages between servers using XML-RPC protocol, but the second version released in 2003 adopted the SOAP protocol [6, pp. 628–629]. One year later, third version was developed due to need of more advanced user administration system. Fourth version of X-Road which contained improvements in the security of data exchange was released in 2006 and remained as the main version of structure in large information systems in Estonia for around four years. In 2010 fifth version was released as result of developments in information technology, e.g., the style of WSDL was switched from RPC/Encoded to Document/Literal wrapped.

In 2012, Cybernetica AS, the creator and maintainer of X-Road versions 1 to 5, started a product development project with the goal of creating a version of X-Road that could be deployed outside Estonia [7]. Based on the product prototype, Cybernetica created X-Road version 6 that is the currently active X-Road version used in Estonia and Finland. Continuing the product development track, the prototype has evolved to Cybernetica UXP® (Unified eXchange Platform) product that targets export markets. For example, UXP has been implemented in country level in Greenland and Ukraine [8], but also, the system has been adapted in an intraorganizaitonal set up in Japanese trust-bank [9]. In 2018, X-Road development was taken over by NIIS (Nordic Institute of Interoperability Solutions, a non-profit established by Estonian and Finnish governments) [10] whereas Cybernetica has continued the UXP development and used it as the basis of several e-government solutions. Although the two systems are developed by separate organizations with separate requirements, they share the same roots and the same basic working principles.

1.1.1 UXP Concepts

Generally, a installation of UXP system (**UXP instance**) contains of the following parties in the broader sense [11]:

- **Governing Authority** is an organization that administrates the whole UXP installation, i.e., establishes security policy and provides technical support for UXP

members.

- **UXP Members** are the individuals (e.g., organizations) which would like to exchange information with each other. It is assumed that each member has an information system (i.e., Subsystem in UXP infrastructure) that is connected with another member's information system.
- **Trust Services Providers** are the entities that supply **certification** and **timestamping services**.

However, in the technical view, the parties consist of the following components [11]:

- **UXP Registry** is a server that is under management of Governing Authority. This component stores information about UXP Members and their UXP Security Servers. Moreover, it stores the security policy of the installation and distributes **global configuration** to the security servers of all members. Concepts of the UXP Registry are further examined in the upcoming chapter.
- **UXP Security Server** is used to connect UXP member's information systems to the UXP infrastructure. It is responsible for forwarding the messages securely between the UXP members. All the messages signed, logged and timestamped by the UXP security servers to ensure their proof value.
  - **Management Services' Security Server** is a security server associated with the UXP Registry that provides management services for the other security servers in UXP instance.
- **UXP Service** is a service provided over UXP infrastructure. Communication is implemented as standardized SOAP or REST (Representational state transfer) web services. This ensures that organizations do not need extensive software development to join the UXP platform.
  - **Service Provider** is a UXP member or subsystem that provides service over UXP infrastructure. Service providers design and implement services and make them available to service clients.
  - **Service Client** is a UXP member or subsystem that uses services provided by UXP service providers over UXP infrastructure.
  - **Management Services** are services that are mediated by the UXP Registry using the Management Services' Security Server. These services are called by other security servers in UXP instances to register changes in configuration made by the security server administrator (e.g., registering new subsystems).
- **Subsystem** represents a UXP member's information system that must be declared by UXP members to consume or provide UXP services.

The components of UXP in an example installation are provided in the Figure 1.

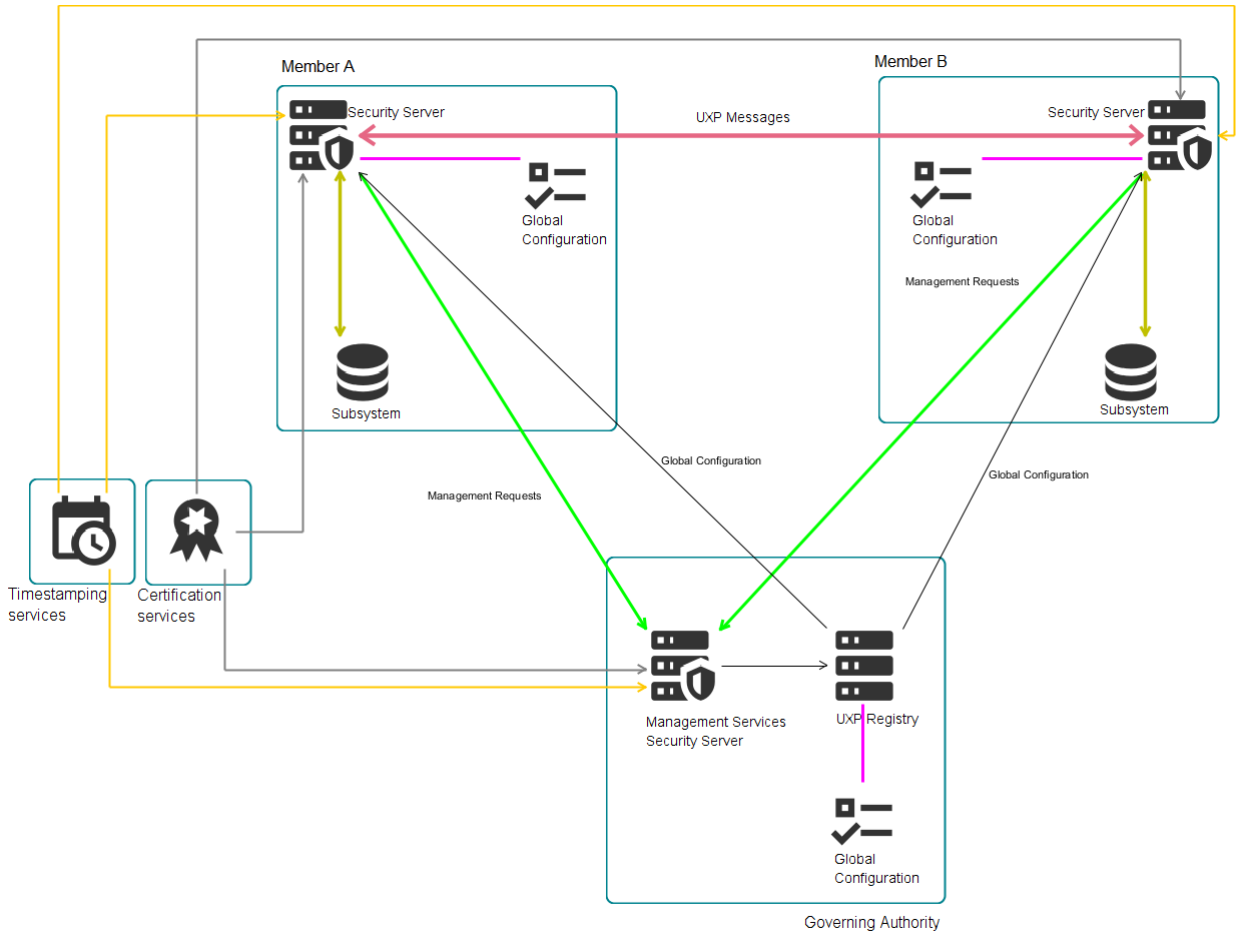

Figure 1. UXP Architecture

The communication is UXP bases on synchronous service calls [11]. It is a task of service providers to design UXP services and grant access rights for the service clients. To make the service available to the service client, both parties have to enter into agreement that specifies the following attributes:

- terms of the service;
- SLA (service level agreement) by the provider;
- security requirements that the service client must meet.

UXP allows members to communicate without intermediates [11], that means, all the information is exchanged directly over encrypted and mutually authenticated channel. It is important to note that Governing Authority does not participate in the actual information exchange, i.e., contents of the messages are not revealed. The TLS (Transport Layer Security) protocol is used to set up a secure channel between security servers. Mutual certificate based authentication is used as well, that means, service provider and service client have to present a valid certificate that is registered by Governing Authority. To

illustrate the process of message exchange between members in the initialized UXP instance, the Figure 2 is presented. In the figure, the steps that require valid information from the global configuration provided by the UXP Registry are explicitly outlined. In other words, UXP members are not able to communicate with each other using UXP infrastructure if the global configuration provided by the UXP Registry is not available and/or is corrupted.

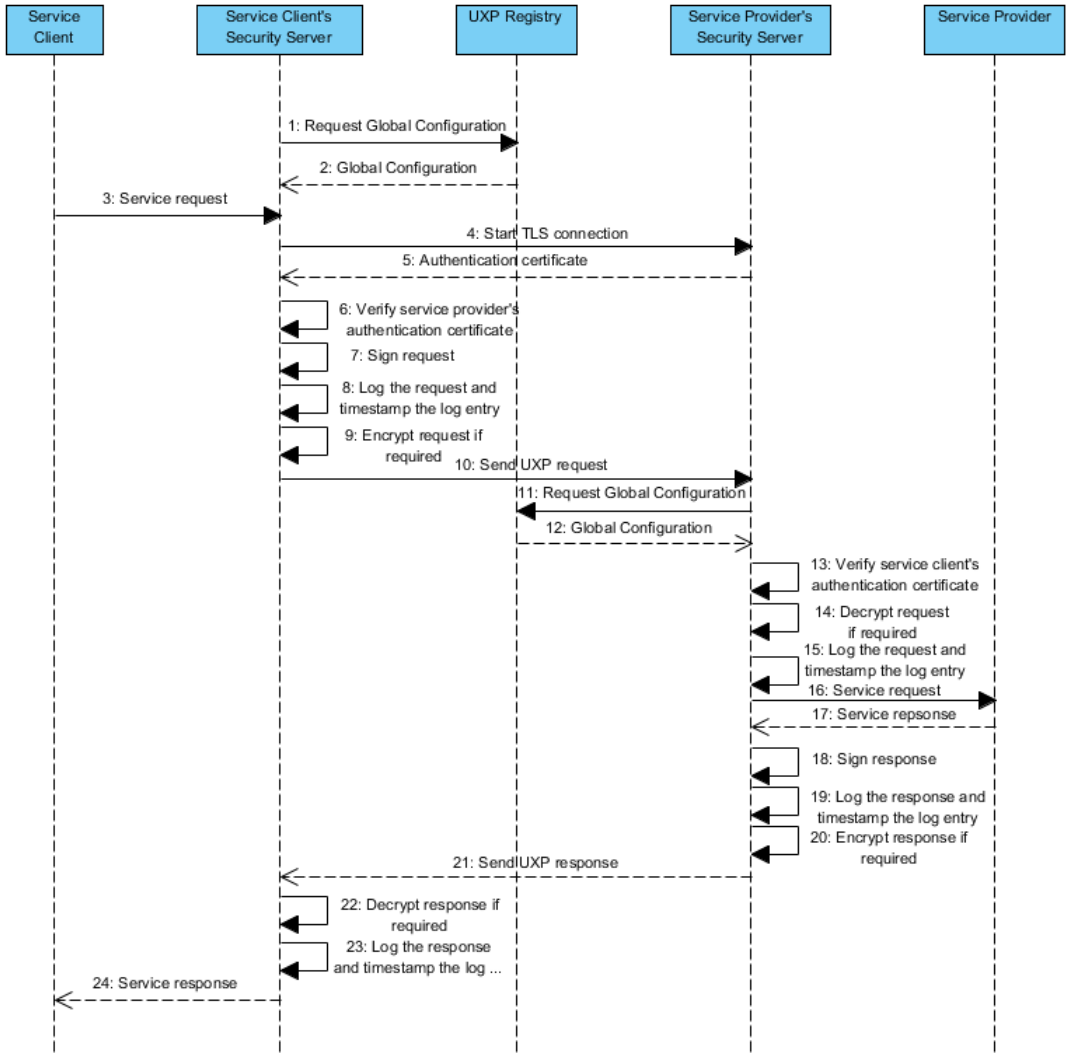

Figure 2. Simplified Process of UXP Message Exchange

The UXP provides various supplementary components that support carrying out specific processes that are out of scope this thesis:

- **UXP Monitoring Server** collects and presents performance and statistical information from security servers. This server is necessary in UXP to discover and debug problems in the infrastructure. Also, it helps to discover usage that does not follow usual patterns, i.e., it is possible to discover potential unauthorized use of services [11].

- **UXP Connector** is an optional application to ease the implementation of UXP services. It is a interface between UXP security server and member's information system's SQL (Structured Query Language) database [12]. The concept of the Connector is that security server sends a service request to the Connector where the request is converted to an SQL statement. Then this request run on a database and the response forwarded to the Connector. The response is transformed into UXP message and sent back to the initial security server.
- **UXP Portal** is an optional component in UXP that provides a universal user interface for consuming UXP services [12]**.** The Portal uses UXP security server to download a list of all UXP members that provide services in the corresponding installation. Then the services are imported to the Portal using their WSDL (Web Service Description Language) descriptions. According to the service description, portal dynamically creates the input form which can be used by users to conveniently consume the services.
- **UXP Directory** is an optional component that provides an overview of all UXP installation's members, information systems and services [11]. Similarly to the UXP Portal; members, information systems and services are downloaded via security server and provided in a user-friendly web interface or REST API (Application Programming Interface) .

1.2 Research Problem

The UXP system provides a data exchange infrastructure where instead of a centralized registry the parties exchange information directly with each other. Although, the management in the UXP is coordinated centrally by the UXP Registry. The registry server is maintaining the list of members and distributes it among the members of the infrastructure [11]. In every instance of the UXP, there exists one central agency that is responsible for establishing communication standards and security policies. However, that brings another challenge that the UXP Registry acts as a trusted third party: all the members have to trust that the party does not misuse its obligations [13]. Due to centralized data storage, another issue arises that an attacker may take unauthorized control over the database and cause damage to the parties that need to access to the valid data. For example, if the information about valid certification authorities and members stored in the UXP Registry is tampered with, the invalid data is made available to the security servers and thus, the secure communication between UXP members will be disturbed.

One way to reduce the risk that the communication between UXP members is interfered because of the centralized management, is to replace UXP Registry as trusted third party with a distributed ledger [13]. One of the distributed ledger technologies is blockchain:

a data structure consisting of linked records that are generated after designated time or event [14]. Despite the popular usage of blockchain in a cryptocurrency, it is further used in smart contracts as well, i.e., validating other types of contracts**.** One influential feature of blockchain is the data integrity [15] which is well consistent with the concept of the UXP [11].

According to the need to improve the existing model of UXP Registry to reduce the need to trust to the correct behavior of the Governing Authority and whilst considering the advantages of new technologies, the research questions were formulated according to the Trivium method, i.e., grammar, formal logic and rhetoric [16]. The fundamental question which is presented below is divided into three sub-questions that are followed with the grammar questions:

- How to reduce the need to trust the correct behavior of the Governing Authority that manages the UXP Registry?
  - RQ 1: How do the requirements for UXP Registry influence the design of the improved UXP Registry?
    - Q 1.1: What are the existing functions of the UXP Registry?
    - Q 1.2: What are the current business processes of the UXP Registry?
    - Q 1.3: What kind of data must be stored in the UXP Registry?
  - RQ 2: How can blockchain technology reduce the need to trust the correct behavior of the Governing Authority?
    - Q 2.1: What are the attributes of blockchain?
    - Q 2.2: What are the features of blockchain that can be used to fulfill the requirements of the UXP Registry?
    - Q 2.3: What are the features of blockchain that can be used to increase the transparency in UXP management?
  - RQ 3: How to design new model of the UXP Registry that uses blockchain technology?
    - Q 3.1: What are the existing blockchain implementations that can be taken into account when developing new version of the UXP Registry?
    - Q 3.2: What is the proposed architecture for improved UXP Registry?
    - Q 3.3: What are the prospective disadvantages of the proposed solutions?

1.3 Research Methodology

The aim of this research is to analyze the perspective of the usage of blockchain in UXP Registry. Further, the aim of this research is to verify whether the application of blockchain is consistent with the requirements of the UXP Registry and would improve the security

properties. Thus, the considered research technique is the design-science methodology. This is because it meets the corresponding requirements [17, p. 83], e.g., the aim is to produce a viable artifact, the objective is a technology-based solution to a relevant business problem and the research is planned to be presented to the technology- and business-oriented public.

During the study, it is planned to examine the reasons behind of the UXP Registry architecture model. That includes analyzing the present documentation of UXP Registry and processes within the system. In addition, the existing literature of blockchain and its applications on diverse registries will be evaluated to discover feasible appliances.

1.4 Structure of the Thesis

In the first chapter, introduction to the topic is presented by describing the background of UXP, research problem and the used methodology. In the second chapter the main concepts of UXP Registry are analyzed further and its processes and weaknesses are outlined. The third chapter introduces the main characteristics of the blockchain technology and its applications with explanations how the features of existing solutions were improved are summarized. In the fourth chapter, the analysis of UXP Registry and blockchain applications is evaluated and the appropriate solution to improve UXP Registry is introduced. In the final chapter, conclusion of the thesis is provided.

2 UXP Registry

The purpose of this chapter is to give an overview of the UXP Registry's role in UXP instance and describe its use cases in general. The main role of the UXP Registry server is to manage the lists of UXP members and security servers [11]. Furthermore, it maintains the list of trusted certification and time-stamping authorities. This info is put together into global configuration and made available to the security servers. To maintain a high-availability of the system, UXP Registry can also be installed in cluster to replicate database between the nodes [18]. It is expected that the Governing Authority maintains the appropriate security conditions (e.g., firewall and other inter-organizational measures) where unauthorized persons cannot access UXP Registry's database.

2.1 Trust Services Management

In UXP, organizations' information system exchange data directly with each other [11]. To ensure that the parties are mutually authenticated and the exchanged messages are usable as evidence in courts, the qualified certification services are essential to prove the ownership of a public key. There are three different types of certificates used in UXP infrastructure:

- authentication certificates to establish secure communications channels;
- signing certificates to digitally sign exchanged messages;
- encryption certificates to encrypt exchanged messages in addition to the encryption provided by the TLS protocol.

Furthermore, timestamping services are used "to prove the existence of certain data before a certain point of time without the possibility that the owner can backdate the timestamps" [19]. Certification services and time-stamping services are part of a security policy that is defined by the Governing Authority and managed using the functionalities provided by the UXP Registry [11].

Using the web user interface, the administrator who has the access rights performs the following processes [19]:

Use Case 1: Manage approved certification services

- Actor:
  - Registry administrator
- Preconditions: -
- Postconditions:
  - Approved certification service is updated.
- Main Success Scenario:
  1. Registry administrator whether:
     (a) enters new,
     (b) edits existing or
     (c) deletes the approved certification service.
  2. System validates and stores the entered information.
  3. System logs the action to the UXP Registry audit log.

Use Case 2: Manage timestamping services

- Actor:
  - Registry administrator
- Preconditions: -
- Postconditions:
  - Timestamping service is updated.
- Main Success Scenario:
  1. Registry administrator whether:
     (a) enters new,
     (b) edits existing or
     (c) deletes the timestamping information.
  2. System validates and stores the entered information.
  3. System logs the action to the UXP Registry audit log.

In addition, supplementary processes can be carried out to view the details of the registered certification services, timestamping services and their details.

2.2 Member Management

Maintaining the list of organizations that use the UXP infrastructure to communicate with each other, is a responsibility of the Governing Authority [11]. UXP Registry provides the following functionalities for administrators to manage the list of members and other UXP components that are necessary for communication over UXP infrastructure [20], [21].

Use Case 3: Manage UXP Members

- Actor:
  - Registry administrator
- Preconditions:
  - At least one member class has been configured.
- Postconditions:
  - Member information is updated.
- Main Success Scenario:
  1. Registry administrator whether:
     (a) enters new,
     (b) edits existing or
     (c) deletes the member information.
  2. System validates and stores the entered information.
  3. System logs the action to the UXP Registry audit log.

To group similar UXP members within UXP instance, member class attribute is implemented in UXP Registry [11]. For example, in Estonia class "GOV" is used to distinct public organizations in X-Road instance [22].Thus, UXP Registry supplies the operations to be carried out by the authorized administrator stated below [21]:

Use Case 4: Manage member classes

- Actor:
  - Registry administrator
- Preconditions: -
- Postconditions:
  - Member class information is updated.
- Main Success Scenario:
  1. Registry administrator whether:
     (a) enters new,
     (b) edits existing or
     (c) deletes the member class information.
  2. System validates and stores the entered information.
  3. System logs the action to the UXP Registry audit log.

To simplify service access rights in the instance level services, global groups can be set up to facilitate the management of UXP subsystems that use the same services. Therefore, the following operation is provided in UXP Registry [23]:

Use Case 5: Manage global groups

- Actor:
  - Registry administrator
- Preconditions: -
- Postconditions:
  - Global group information is updated.
- Main Success Scenario:
  1. Registry administrator whether:
     (a) enters new,
     (b) edits existing or
     (c) deletes the global group information.
  2. System validates and stores the entered information.
  3. System logs the action to the UXP Registry audit log.

Similarly to the trust services management, details of the registered UXP members, global groups and member classes can be viewed using the corresponding functions provided by the UXP Registry.

2.3 Security Server Management

Maintaining the list of UXP Members' security servers and related components, is another responsibility of a Governing Authority [11]. UXP Registry provides the following functionalities for administrators to manage the list and settings [20], [21].

Use Case 6: Manage UXP Member's security servers

- Actor:
  - Registry administrator
- Preconditions:
  - UXP Member has been registered in UXP Registry.
  - Either:
    - security server has been registered in UXP instance (in case of editing existing information or deleting the security server) or
    - security server administrator has sent a security server registration request (in case of registering new security server).
- Postconditions:
  - UXP Member's security server information in UXP instance has been updated.
- Main Success Scenario:

1. Registry administrator whether:
   (a) enters new,
   (b) edits existing or
   (c) deletes the security server information.
2. Registry administrator uploads the authentication certificate (in case of registering new security server).
3. System validates and stores the entered information.
4. System logs the action to the UXP Registry audit log.

Use Case 7: Manage UXP Member's security server's authentication and encryption certificates

- Actor:
  - Registry administrator
- Preconditions:
  - UXP Member has been registered in UXP Registry.
  - Either:
    - UXP Member's security server has been registered in UXP instance; or
    - UXP Member's security server administrator has sent a security server registration request (in case of registering new security server).
  - Security server administrator has sent a certificate registration or deletion request.
- Postconditions:
  - UXP Member's security server certificate information in UXP instance has been updated.
- Main Success Scenario:
  1. Registry administrator whether:
     (a) uploads new
     (b) or deletes the existing security server authentication or encryption certificate.
  2. System validates and stores the entered information.
  3. System logs the action to the UXP Registry audit log.

Use Case 8: Manage UXP Member's subsystems

- Actor:
  - Registry administrator
- Preconditions:
  - UXP Member has been registered in UXP Registry.

- Postconditions:
  - UXP Member's subsystem information in UXP instance has been updated.
- Main Success Scenario:
  1. Registry administrator whether:
     (a) enters new
     (b) or deletes the existing UXP Member's subsystem information (except if the subsystem is registered as a security server client).
  2. System validates and stores the entered information.
  3. System logs the action to the UXP Registry audit log.

Use Case 9: Manage UXP Member's security server client

- Actor:
  - Registry administrator
- Preconditions:
  - UXP Member has been registered in UXP Registry.
  - UXP Member's security server has been registered in UXP Registry.
  - Either:
    * security server client has been registered in UXP instance (in case of editing or deleting existing security server client) or
    * security server administrator has sent a security server client registration request (in case of registering new security server client).
- Postconditions:
  - UXP Member's security server client information in UXP instance has been updated.
- Main Success Scenario:
  1. Registry administrator whether:
     (a) enters new
     (b) or deletes the existing UXP Member's security server client information.
  2. System validates and stores the entered information.
  3. System logs the action to the UXP Registry audit log.

Similarly to the trust services management, details of the registered security servers, subsystems and their certificates can be viewed using the corresponding functions provided by the UXP Registry.

2.4 Configuration Management

The main role of UXP Registry is to distribute configuration (i.e., approved trust services and registered UXP members) to the security servers of all members [24]. After a designated time, system generates a configuration files that are made available using configuration anchor, i.e., information that is used by the security servers to access the configuration source and verify the configuration. To ensure that the configuration maintains its authenticity, it is signed by the private key of UXP Registry. The configuration contains the information that is required for establishing UXP communication between security servers, i.e., the information that has been specified in the previous sections 2.1, 2.2 and 2.3.

Besides, as all the information that is necessary to generate the configuration is maintained centrally in the UXP Registry, backup functionality is provided to ensure that the UXP Registry could be efficiently restored after malfunctions [20].

Use Case 10: Distribute global configuration

- Actors:
  - UXP Registry
  - Security server
- Preconditions:
  - System configuration contains the necessary data.
- Postconditions:
  - Security server has received global configuration.
- Main Success Scenario:
  1. UXP Registry generates the configuration files from the system configuration.
  2. UXP Registry signs the configuration files with its private key.
  3. Security server requests valid configuration from the UXP Registry.
  4. UXP Registry provides the valid configuration files to the security server.
  5. Security server verifies that the signature of the configuration files is valid.
  6. Security server verifies that the configuration files are valid.
  7. Security server stores the configuration files.

Use Case 11: Backup UXP Registry

- Actors:
  - Registry administrator
- Preconditions: -

- Postconditions:
  - Back up file has been created.
- Main Success Scenario:
  1. Registry administrator selects to back up the UXP Registry configuration.
  2. System creates the backup file that contains the system configuration.
  3. System stores the entered information.
  4. System logs the action to the UXP Registry audit log.

Use Case 12: Restore UXP Registry

- Actors:
  - Registry administrator
- Preconditions:
  - Back up file has been created.
- Postconditions:
  - UXP Registry configuration has been configured according to the backup file.
- Main Success Scenario:
  1. Registry administrator selects to restore the UXP Registry configuration.
  2. System validates the backup file.
  3. System restores the configuration from the backup file.
  4. System logs the action to the UXP Registry audit log.

The processes described in the previous sections present that the UXP Registry has a major role in maintaining the policy that is agreed by the members of UXP instance, e.g., which trust services are allowed to be used to assure a secure channel between the members to exchange UXP messages. That confirms that the UXP Registry upholds a role of the trusted third party, in other words, all the members have to trust that the party does not misuse its power and provides its services conscientiously [13]. Because of this data is maintained centrally (in the best case scenario, high-availability is used to reduce the accompanying risks), another issue arises that an attacker may gain control over the central database and invalidate its assurance to provide the necessary configuration for UXP members.

Currently, the opportunities to control whether the UXP Registry has been behaved according to the agreements are limited: although the most actions made by the Registry administrator are logged in audit log, the access to the logs is not publicly available for the members. Furthermore, the actions maintained in the audit log do not reflect the actual reasons behind the actions, rather they simply state the fact of the action. That means, members would have to request external audit to determine whether the Governing Authority has been acted according to the agreements. This is why the lack of transparency

and the threats that accompany centralized registries has led various registries to implement distributed ledgers, i.e., blockchain technology to eliminate these issues. The main difference between using a third party and a distributed ledger system is that the ledger eliminates the need to trust any particular party [13].

3 Blockchain Technology

3.1 General Concepts

Blockchain is a recent discovery in secure computing that provides decentralized authority in an open networked system [14]. The main concept of blockchain is to replace the centralized database with authoritative access control. It is created and maintained as a hierarchical and chronologically-ordered chain of blocks with timestamp since its inception in 2009 when the Bitcoin was launched. It allows the communication between untrusted parties without the existence of a middle man [25].

Blockchain is often defined as a distributed ledger [15]. A ledger is a data structure where the transactions are formed into an ordered list, e.g., monetary transactions between multiple financial institutions. In general, blockchain is a distributed database that logs an evolving list of transaction records by organizing them into a hierarchical chain of blocks. Each block stores a list of records and is chained to the previous one, by including the hash value of it [26].

There are several attributes that the data layer includes [27]: block, structure of the chain, hash function, Merkle tree, timestamp, asymmetric encryption etc. Block, in turn, consists of head and body. In the block head, the address of the previous block and the target hash value of the current block, are included. Also, the system encloses all security transaction records according to security requirements in the block. Every blockchain starts with a special block, called genesis [26], which does not reference a previous block and must be known from a start by all the peers.

There are two fundamental building blocks for implementing the blockchain [14]:

- **Hash pointer** is a cryptographic hash of the data that points to the location in which the data is stored. Therefore, it is possible to use the hash pointer to check whether the data has been tampered with or not, because they are used to link data blocks together. In case that an attacker attempts to change data in any block in the whole chain, the hash pointers of all previous blocks have to be changed. Ultimately, the adversary has to stop tampering because he will not be able to falsify the genesis

that has generated once the system has been constructed.

- **Merkle tree** is an additional structure used for building blocks. It is a binary search tree where tree nodes are linked to one another using hash pointers. Nodes are grouped separate at joints, i.e., each time two nodes at the lower level are grouped into one at the higher level. Thus, new data node is created that contains the hash value of each. This process is repeated until the root of the tree is reached. It prevents data from tampering by crossing down through the hash pointers to any node in the tree, i.e., when an attacker tries to tamper data at a leaf node, hash value is changed in its parent node. That means, all nodes on the path from the bottom to the top need to be changed, but the initial value of genesis is known to all parties and thus, the attack will be revealed. Block head must be time-stamped [27] to indicate the time of this transaction for ensuring the orderly arrangement of security blocks.

A typical blockchain system consists of multiple nodes that do not completely trust each other [15]. "Together, the nodes maintain a set of shared, global states and perform transactions modifying the states. All nodes in the system agree on the transactions and their order." All nodes maintain an identical chain of blocks at the same time and do not rely on central authority to keep malicious adversaries from disrupting the coordination process of reaching consensus [14]. Once some data has been recorded into the global ledger blockchain, it is impossible to change the blockchain, and by enforcing the majority agreement of update validity through consensus, it ensures the consistency state. Smart contracts are typically distributed protocols that execute formalized contract terms autonomously, therefore, risk of human error and manipulation is reduced [13]. This always opens an opportunity to automate processes between two or more participants without trusted third parties.

The blockchain is maintained by a set of members who participate to the consensus protocol [26]. The transactions that are not added in a block yet, are collected by members who eventually construct a new block that may differ for everyone. Usually, when this new block reaches a predefined maximum size or timer expires, a distributed consensus protocol starts. That means, as a result, one member is elected as temporary central manager and decides which block will be added to the blockchain next. The temporary manager signs the block and broadcasts it to all the parties so they can verify whether the block was built from valid records or not, and append it to their locally maintained blockchain. The block header included in this block contains all the information that is needed to verify the correctness of the executed consensus protocol. When a new block is sent to the network, each node has the option whether to add that block to their copy of the global ledger or to ignore it [14]."The consensus is employed to seek for the majority of the network to agree upon a single state update in order to secure the expansion of the global ledger (the

blockchain) and prevent dishonest attempts or malicious attacks." The structure of a typical blockchain is displayed in the Figure 3.

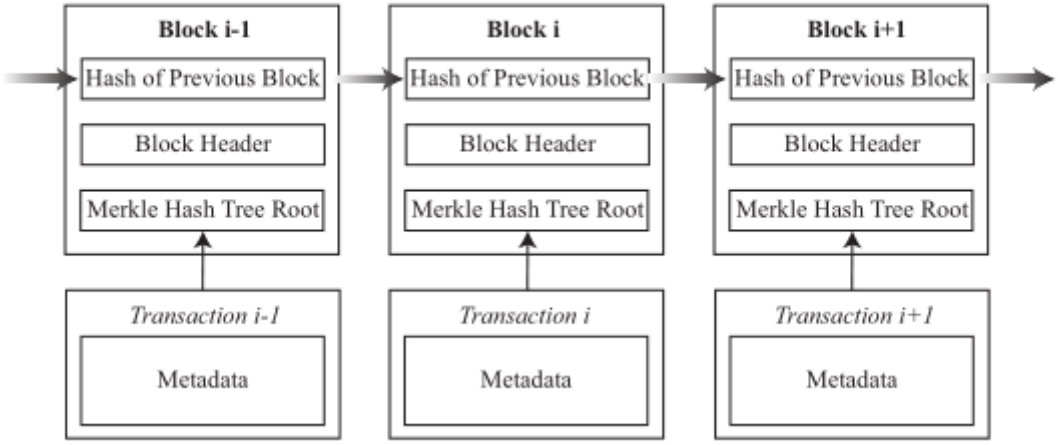

Figure 3. Typical Blockchain Structure [1]

Blockchains can be generally divided into two: public (i.e., permissionless) or private (i.e., permissioned) [14], [26]. In private blockchain, records can be written only by single authorized party, also, a consortium type of blockchain can be distinguished where the write privilleges have been extended to a group of organizations. This is beneficial when records contain privacy-sensitive or business-critical information. Maintaining copies of sensitive data in all the peers involved in a distributed ledger could be highly costly. Furthermore, peers must be constantly online to perform consensus protocol for each block of data to be recorded in the ledger. A distributed ledger is maintained in a distributed fashion, and it does not need for central administration or centralized data storage. A consortium member can be a client or a peer: the clients can only read and write records on the ledger, but the peers are in addition in charge of executing the consensus protocol. A private distributed ledgers can securely carry personal and critical information, since only the authorized entities can access it.

In contrast to the private, in the public blockchain, all the information related to the transactions are public to any interested party anywhere [28]. Anyone in the world can take part in such a system with no restrictions being placed on when users join or leave the system. These characteristics of public blockchain systems have led to scaling problems and bad performance compared to traditional transaction processing systems. It makes it difficult to control information sharing and obey privacy regulations like GDPR (General Data Protection Regulation). As cryptocurrencies generally use public blockchains, concomitant anonymity has also led people to various types of illegal activities. Additional issue is with the anonymous people who started DoS attacks in the network,

therefore, consensus protocols like PoW (Proof of Work) that were invented to deal with such issues.

3.2 Blockchain Applications

In addition to the initial secure properties of blockchain that were addressed in the previous section, various researches have been conducted to discover and use blockchain attributes to resolve issues related to centralized databases.

Rouhani and Deters have been analyzing various access control systems where blockchain was found to be trustable alternative, because the distributed nature of blockchain eliminates the issue for the single point of failure and other centralized management problems [29]. Also, because third parties are eliminated, there is no need for concern about privacy leakage from third parties. Also, due to reason that blockchain logs all the transactions, trustable and unmodifiable history logs are created and can be accessed. By using smart contracts, access permissions under complex conditions can be enforced. Because of applied consensus mechanisms, only valid transactions are recorded on blockchain. They also pointed out that Hyperledger Fabric provides improved lighter consensus mechanism and efficient transaction processing flow, but generally performance of the blockchain-based solutions is not comparable with the existing centralized solutions. It was also recommended that permissioned blockchain platforms should be used in access control systems, because they support transaction privacy and private data that is not achieved in public blockchains. It was also detected that blockchain is not a suitable for storing a big volume of data, therefore, the data must be stored in secure off-chain storage and the access policies, the hash of the data, and references to the data record on blockchain. The secure integration between on-chain and off-chain can bring challenges.

Another access control based study was carried out for plant phenotyping system [30]. There was pointed out that blockchain technology could provide provenance data, meaning that information of every single transaction is recorded in the blockchain network. Also, the trust in system can be increased because:

- data is replicated towards the blockchain network which makes it available to the participants even if the network is partially not available;
- and all the transactions are validated by the entire blockchain network.

Du et al. analyzed application of blockchain in the security system and identified that a blockchain technology makes security data traceable, untouchable and more transparent [27] because in the centralized management, it is simple to tamper data, hardware

technologies solutions are behind time and personnel in responsible for data may be unreliable.

Zhangm Xue and Liu evaluated various security and privacy attributes on blockchain [14]. For online transactions, it is outlined that blockchain provides consistency - all nodes have the same ledger at the same time, that makes sure that data of each entry at each node of the system gets consistent eventually, and thus achieves high availability and low latency at the risk of returning stale data. What is more, tamper-resistance to any type of intentional tampering to a data maintained in blockchain has been identified. It means that any transaction data stored in the blockchain cannot be tampered during and after the process of block generation, because multiple cryptographic techniques are used [15], [31]. Resistance to a DDoS (Distributed Denial-of-Service) attack is recognized, because due the consensus protocol in Bitcoin system, processing of blockchain transactions can continue even if several blockchain nodes are unavailable. Bitcoin is resistant to double-spending attack as it evaluates and verifies each transaction using the transaction logs in its blockchain with a consensus protocol. In addition, transaction are signed by its sender which ensures that if someone alters the transaction, it can be easily detected. However, it was outlined that performance cost is too high to be affordable for all cases. High operations cost was also highlighted by Arena, Perazzo and Dini where it was indicated that in private distributed ledgers operational cost is generally quite expensive, since all the peers have to maintain the entire record database [26]. In cryptocurrencies it has to be taken into account that loss of the private keys has direct and irreversible financial impact [15].

Abdelhamid and Hassan have addressed that codifying, security, privacy, availability and performance issues of Smart Contracts need to be addressed in future studies [25]. Lack of studies on criminal activities and deploying Smart Contracts on different Blockchain platforms other than Ethereum [25].

3.2.1 Blockchain Applications on Registries

Because of the attributes of blockchain, several fields can benefit from its merits. “For example, in healthcare, the blockchain could help to create immutable audit trails, maintain the reliability of health trials, and uphold the integrity of patient data.” [14]. Further, data insertion has been one of the first valuable applications of the blockchain beyond the cryptocurrencies [31].

Several researches have studied how to decentralize registries using distributed ledger technologies. In Georgia, property registration system was developed with the custom-

designed blockchain technology [32, pp. 673–674]. The architecture of the solution contains of the following steps: at first, application is filed which is followed by the document generation in the system. Next, using the certificate, document is signed with a timestamp and a hash of the document is generated. The hash is sent to the Bitcoin networks which completes the transaction. The status information of the all steps can be inquired using the designated websites.

In Brazil, development of the SWIM (System Wide Information Management) Registry for air traffic management with the blockchain support has been presented [33, pp. 3544–3545, 3549]. The SWIM is a layer-based interoperability framework and an oriented-service environment which allows to represent and define exchanges of data between authorized partners. Stakeholders of SWIM publish and expose services for consumers, whereas the communication is implemented between interconnected registries. The key element of SWIM is the SWIM Registry which manages the interoperability features with implementation of access control application. Further, it references to the available services with their usage conditions and contains other relevant information about the system, e.g., models of information exchange, policies and infrastructure. In the research, algorithms based on blockchain were introduced for SWIM Registry Brazil flight plan service. The solution demonstrated the beneficial attributes of blockchain such as user verification and authorized access. Besides, the flight plan service was improved in general as a result of better access to the information by the users.

Margheri et al. [34, pp. 688–690] have represented the usage of blockchain system for FaaS (Federation-as-a-Service) infrastructure. The FaaS project was initiated because of the proposal of concept of cloud federation, i.e., aggregating different services from various cloud providers in a single pool. Nevertheless, the suggested solution had to take into account that all the members of the federation should be "a network of peers equally concurring to their governance". As blockchain has the appropriate features, the usage of that technology was commenced using smart-contracts to build the federation registry. The guarantees of data integrity and service availability were the fundamental advantages which contributed to the selection of blockchain. The implementation of the following functionalities were provided:

- the storage of business contract and its signature;
- snapshots of the federation state;
- administration actions on access control and service level agreement polices;
- data sharing services;
- the storage and processing of gathered logs.

The architecture of the FaaS infrastructure contains a private blockchain system which is shared by the other federations. The first implementation of the FaaS registry was developed on Ethereum. This solution does not need third party for the basis of federation and democratic control of the business control is maintained. Also, it improves the security and ensures that the services are available. On the other hand, the potential disadvantages are outlined as well, e.g., limitations of speed, computing and scalability [34, pp. 690–691].

The application of INK consortium blockchain which is an extension of HyperLedger Fabric has been inquired by Zhenfeng et al. [1, pp. 25-27, 31-32]. They researched it in a context of DSES (Decentralized Service Eco-System) using the data from ProgrammableWeb.com. Trust issues, security control and cost of maintenance were the main reasons why the option of transformation to the decentralized system was considered. The results of the study displayed that the INK consortium blockchain covers the requirements of DSES and network query performance is feasible for practical use. However, the authors of the research imply that the cost incurred and willingness to adopt the application should be examined further.

Several researches related to blockchain registries have been conducted in the field of IoT (Internet-of-Things). Almadhoun et al. [35, pp. 3, 8] have studied a user authentication scheme using blockchain where Ethereum smart contracts were used for authentication of IoT devices. The study presents the proposed architecture and testing scenarios. Furthermore, security analysis was carried out which revealed that security goals were achieved and the solution is resilient against recognized cyberattacks. Hyperledger Fabric was adopted for the solution proposed by Stanciu [36, pp. 667, 670]. The study acknowledged that load which can be adequately processed in real time in higher level can have a limitation and thus, the architecture of solution using blockchain has to be well considered. Another Ethereum based IoT device management analysis was accomplished by Alblooshi et al. [37, pp. 151, 155–156] which highlighted that the importance of confidentiality has to be considered while selecting the type of blockchain network, i.e., permissioned or permissionless. This is because in the public and permissionless networks the confidentiality goal is not reached due to clear transactions. When choosing the solution to achieve security and privacy in blockchain-based system, there is no single technology that has no imperfections [14]. When a new technology is added to a technology to a complex system, new forms of attacks may occur. Also, there is always need for a compromise when it comes to the security, privacy, and efficiency.

4 Design of the Improved UXP Registry

4.1 Evaluation of the UXP Registry and Blockchain Overview

After analysis of UXP Registry in the Chapter 2 it was determined that the processes can be divided into three:

1. *Individual processes* that are currently carried out solely by the Registry administrator.
2. *Multi-party processes* that require confirmation and/or initiative from another party, e.g., security server administrator.
3. General *management processes*.

In addition, according to the background of UXP described in the Section 1.1, analysis of the UXP Registry conducted in the Chapter 2 and blockchain and its already deployed applications outlined in the Chapter 3, the following requirements are formed to select the appropriate blockchain type for the UXP Registry:

- **Requirement 1:** There are controller organization (Governing Authority) in the present UXP instances who are responsible for managing the UXP instance.
  - The requirement is derived from the Section 1.1.1, use cases 1-9 and 11-12 presented in the Chapter 2 where Registry administrator initiates changes in configuration on behalf of the Governing Authority.
- **Requirement 2:** The Governing Authority has set certain rules who can access and modify the data necessary for UXP Registry's functioning.
  - The requirement is derived from the Section 1.1.1, use cases 1-9 and 11-12 presented in the Chapter 2 where Registry administrator and/or UXP member's security server administrator initiate changes in configuration.
- **Requirement 3:** The Governing Authority has set rules that must be automatically validated, i.e., it must be possible to formulate processes and requirements.
  - The requirement is derived from the Section 1.1.1, use cases 1-12 presented in the Chapter 2 where Registry administrator and/or UXP member's security server administrator initiate changes in configuration, whereas, Registry verifyies that certain preconditions are met before the changes are submitted.

- **Requirement 4:** Data stored in UXP Registry contains business-critical information, thus, must maintain immutability and authenticity;
  - The requirement is derived from the Section 1.1.1, use cases 1-12 presented in the Chapter 2 where Registry stores and provides global configuration to the security servers that is necessary for exchanging UXP messages.
- **Requirement 5:** Joining the UXP instance is generally not available for everybody, e.g., it is often available on a limited network.
  - The requirement is derived from the Section 1.1.1 and Chapter 2 that describe opposing UXP cross-country (e.g., Ukraine) wide set ups vs. implementation within one organization (e.g., trust bank in Japan).
- **Requirement 6:** Economic cost-effectiveness, the hardware cost should not rise significantly, thus, it is not suitable to adopt the consensus mechanism based on arithmetic [27].
  - The requirement is derived from the existing set up of the UXP (see the Section 1.1.1) where UXP members already maintain security servers to enable access to their subsystems and services. Thus, it is not feasible to impose additional obligations to UXP members by increasing hardware requirements considerably.

These requirements are well in line with the attributes of **private** blockhain types. The cases related to the registry operations analyzed in the Section 3.2 had implemented blockchain-based solutions using Hyperledger Fabric and Ethereum platforms. Although they look similar in their concepts (i.e., both allow permissioned network and implementing smart contracts), the main difference is the used consensus mechanism. In Ethereum, PoW algorithm is used in the ledger level [38], whereas Hyperledger Fabric does not require specific consensus arithmetic, rather transaction level agreement protocols can be developed [39]. Based on the used consensus mechanism, Hyperledger Fabric is considered more viable than Ethereum for implementing UXP Registry processes, thus, the Hyperledger Fabric platform is selected to implement improved UXP Registry.

4.2 Improved UXP Registry

4.2.1 Deployment and Initialization

To deploy operations currently managed by the UXP Registry using the Hyperledger Fabric technology, the components provided in the Figure 4 need to be implemented. In this example configuration, it is expected that there are two UXP Members and Governing Authority who is responsible managing the UXP instance.

A blockchain network in the context of Hyperledger Fabric is a "technical infrastructure that provides ledger and smart contract services to applications" [40]. To form a blockchain network, an ordering service needs to be started**.** Ordering service is provided by the **orderer** component that enforces access control for the channel, i.e., manages the peers who can read and write to the channels [41]. In the permissionless blockchains (e.g., Ethereum and Bitcoin), transactions are ordered into blocks using the probabilistic consensus algorithms that guarantee the ledger consistency but where different participants in the network may have a different view of the accepted order of transactions. However, Hyprledger Fabric is a permissioned blockchain and uses ordering service that relies on deterministic consensus algorithms where "any block validated by the peer is guaranteed to be final and correct". The network is configured in accordance with the **network configuration** that grants the administrative rights to the Governing Authority. However, it is expected that all the members of the blockchain network need to access the same information, thus only one **consortium** that will contain all the UXP instance members (i.e., also the blockchain network members) is created.

The key part of the Hyperledger Fabric's blockchain network is a **channel** which is a main communication system that the blockchain members use to communicate with each other. The channel is defined by the **channel configuration** that defines for example, who can add new members to the channel. In the first perspective architecture, it is expected that only Governing Authority can add new members similarly to the current UXP Registry solution. Each member of the channel will have one **peer** that is the component where the copy of **ledger** is hosted. To support operations in the UXP Registry that are currently managed centrally (see the Chapter 2), **smart contracts** need to be developed for these functions.

Then, the smart contracts must be installed on peers and defined in the channel – all the organizations need to approve the definition as well. In this proposition, smart contracts will be developed that contain transactions needed to provide the functionalities. Hyperledger Fabric smart contract defines the transaction logic that manages the business object's lifecycle [42]. Smart contracts mainly add, get and delete values in the world state, i.e., current value of the business object's attributes. Also, with every chaincode, an endorsement policy is associated to indicate which members in the blockchain network must sign a valid transaction initiated by the smart contract. Business objects on which the smart contracts should be developed were identified in the Chapter 2. Smart contracts can be invoked using **client applications,** it is anticipated that the security servers and UXP Registry will act as applications in the context of Hyperledger Fabric. In addition to using the **certification services** to authenticate UXP components and sign UXP messages, the certification services are required in the blockchain network to identify components that

belong to UXP Members and Governing Authorities.

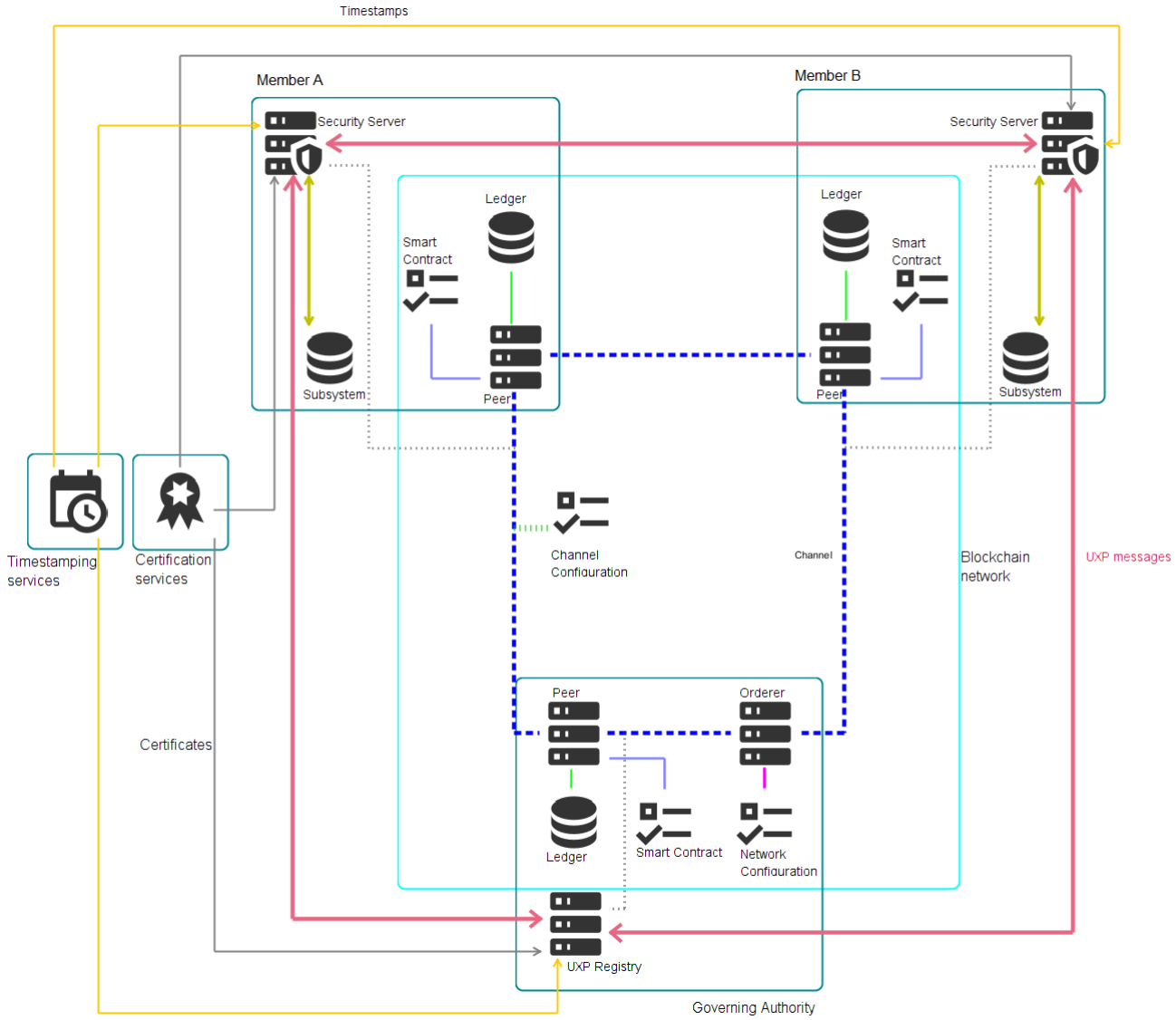

Figure 4. Proposed UXP Architecture Using Hyperledger Fabric Technology

4.2.2 UXP Registry's Current Processes in Hyperledger Fabric

It is expected that the improved UXP Registry, including blockchain network described in the previous section, has been set up. Using the smart contracts functionality provided by the Hyperledger Fabric, the processes highlighted in the use cases described in the Chapter 2 that can be implemented using the smart contracts are described in the Table 1:

Table 1. *Implementing Smart Contracts for Existing Use Cases*

Use case No.	**Use case**	**Process to be implemented using smart contract**
3	Manage UXP Members	Verifying that the precondition has been met: at least one member class has been configured.
6	Manage UXP Member's security servers	Verifying that the precondition has been met: ■ whether security server has been registered in UXP instance; – or security server administrator has sent a security server registration request.
7	Manage UXP Member's security server's authentication and encryption certificates	Verifying that the preconditions have been met: ■ UXP Member has been registered: ■ whether UXP Member's security server has been registered; – or UXP Member's security server administrator has sent a security server registration request. ■ security server administrator has sent a certificate registration or deletion request.
8	Manage UXP Member's subsystems	Verifying that the precondition has been met: ■ UXP Member has been registered.

9	Manage UXP Member's security server client	Verifying that the preconditions have been met: ■ UXP Member has been registered; ■ UXP Member's security server has been registered in UXP Registry ■ Whether security server client has been registered in UXP instance – or security server administrator has sent a security server client registration request.

Steps in the Main Success Scenario that are marked with "*" are initiated only if the process requires endorsement from another party.

Individual and multi-party processes

- Main Success Scenario:
  1. Registry administrator initiates a change in configuration.
  2. UXP Registry forwards the request to the Governing Authority's peer.
  3. * UXP Registry forwards the request to the Member A's peer.
  4. Governing Authority's peer receives the request and constructs a request to invoke a smart contract.
  5. * Member A's peer receives the request and constructs a request to invoke a smart contract.
  6. Governing Authority's peer validates and signs the request.
  7. * Member A's peer validates and signs the request.
  8. Governing Authority's peer invokes the smart contract function and sends a response to the UXP Registry.
  9. * Member A's peer invokes the smart contract and sends a response to the UXP Registry.
  10. UXP Registry validates the responses and broadcasts it to the orderer.
  11. Orderer receives the broadcast, orders it chronologically and creates the block of transactions.
  12. Orderer delivers the blocks of transactions to all the peers, i.e., Governing Authority's, Member A's and Member B's peer.
  13. All the peers validate and append the block to the channel's chain.
  14. All the peers notify the UXP Registry that the transaction has been appended to the chain, i.e., the ledger has been updated.

The corresponding sequence diagram is presented in the Figure 5.

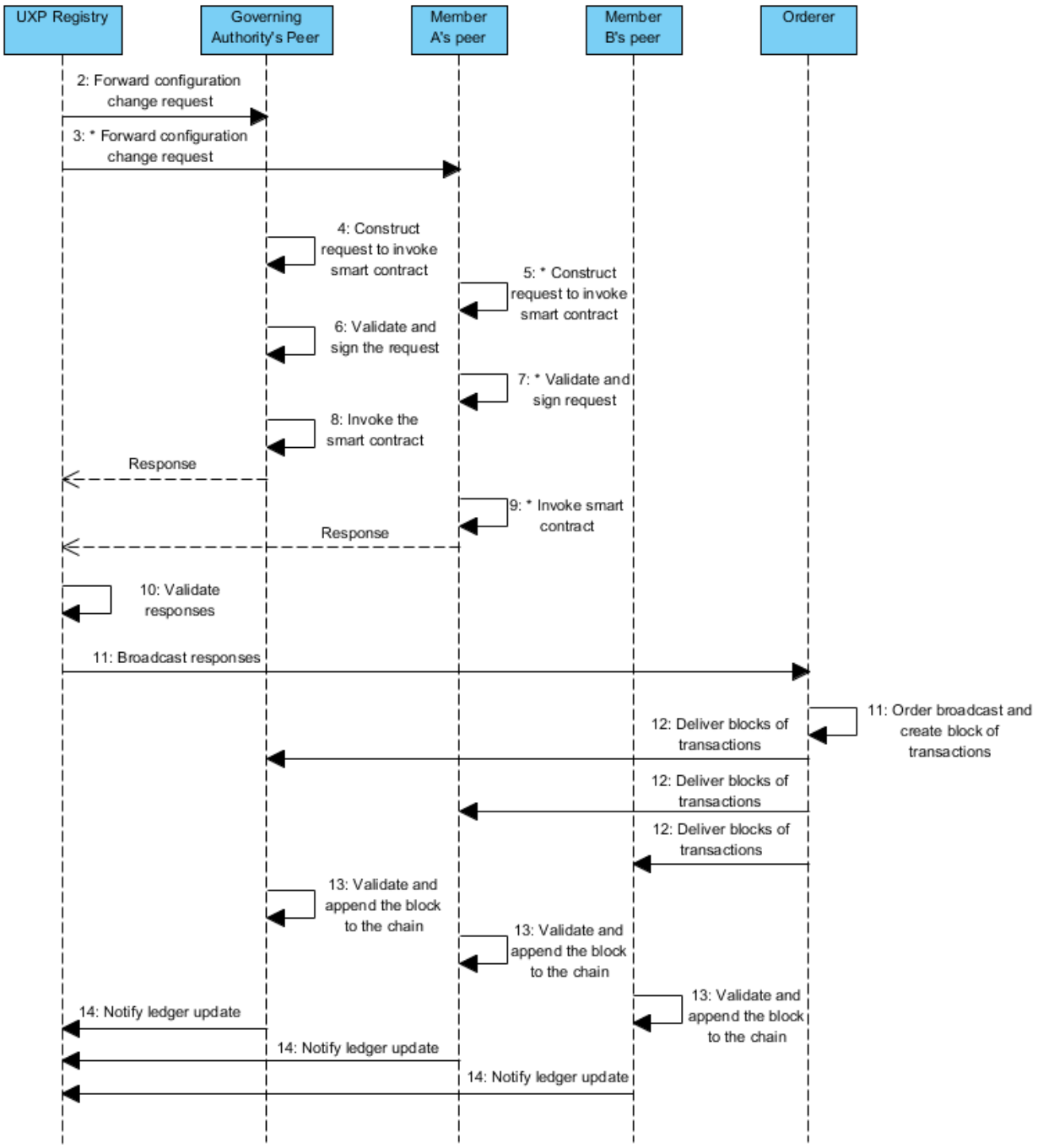

Figure 5. Sequence Diagram of the Individual and Multi-Party Processes

Configuration Management

In the current UXP Registry, configuration is hosted centrally in UXP Registry and made available to security servers through global configuration distribution as described in the Section 2.4. However, with Hyperledger Fabric technology, the configuration in UXP Registry is decentralized and all the peers keep the copy of the ledger that includes the most recent data, i.e., currently valid configuration. Thus, there is no explicit need for the UXP Registry to distribute configuration separately.

When it comes to the backup and restoration, then there is also no specific need to backup and restore, because in case of failure in a peer, it can be started again from the beginning and receive the blocks from other peers and orderer.

4.2.3 UXP Registry's Improved Processes in Hyperledger Fabric

Currently, multiple use cases (6, 7 and 9 described in the Chapter 2) require that security server administrator has sent a request and then Registry administrator approves it manually by initiating a change in configuration. However, if appropriate smart contracts (i.e., formalizing the rules that have to be met) are developed, the current processes can be more automatized. The general use case would look like following:

- **Main Success Scenario:**
  1. Member A initiates a change in configuration.
  2. Member A forwards the request to the Member A's peer.
  3. Member A's peer receives the request and constructs a request to invoke a smart contract.
  4. Member A's peer validates and signs the request.
  5. Member A's peer invokes the smart contract function and sends a response to the Member A.
  6. Member A validates the response and broadcasts it to the orderer.
  7. Orderer receives the broadcast, orders it chronologically and creates the block of transactions.
  8. Orderer delivers the blocks of transactions to all the peers, i.e., Governing Authority's, Member A's and Member B's peer.
  9. All the peers validate and append the block to the channel's chain.
  10. All the peers notify the Member that the transaction has been appended to the chain, i.e., the ledger has been updated.

The corresponding sequence diagram is presented in the Figure 6. This automation would

significantly reduce possible human errors from the Governing Authority's side, because the rules have been agreed on and validated accordingly.

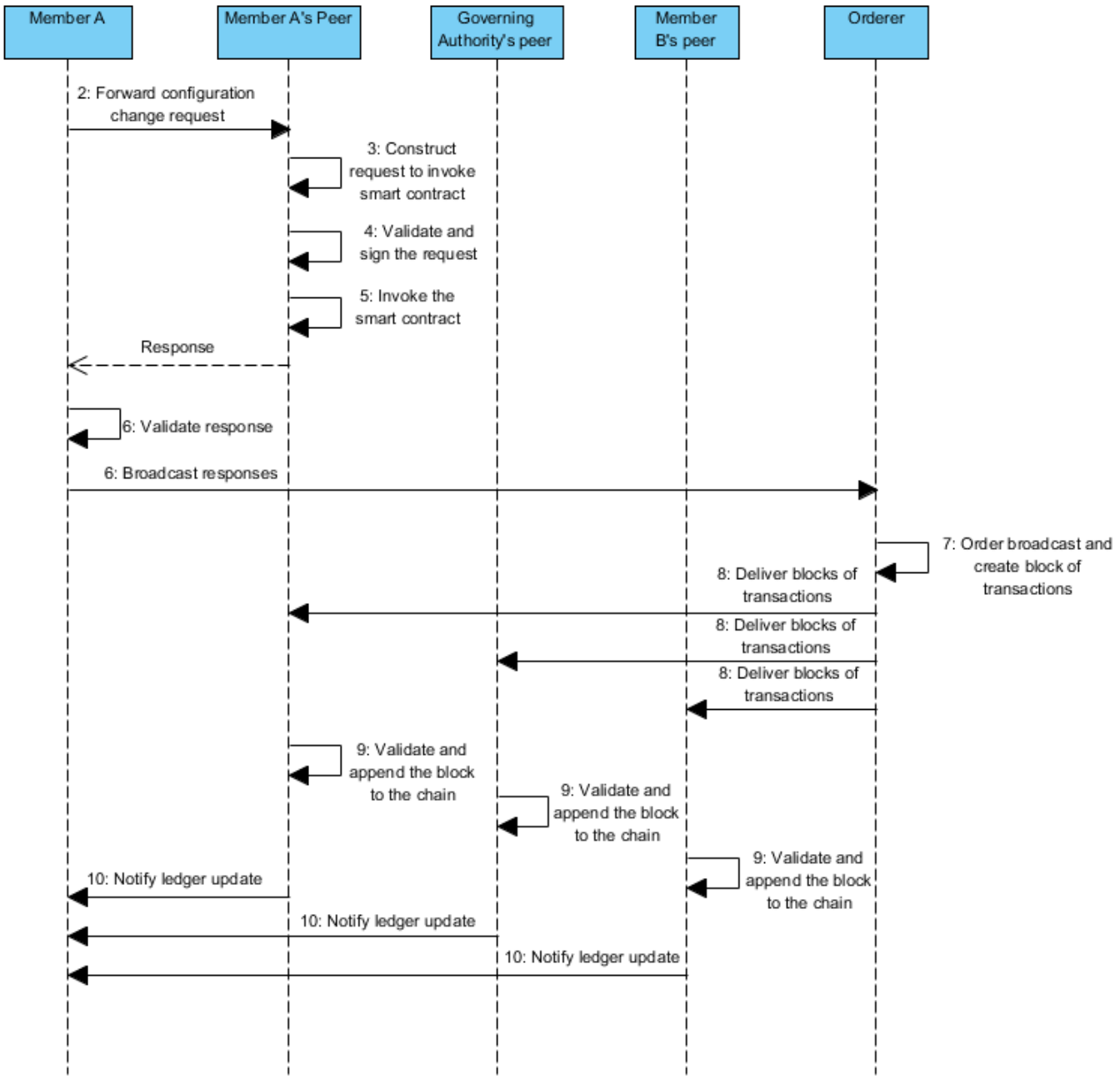

Figure 6. Sequence Diagram of the Improved Process

4.3 Evaluation

Deploying blockchain-based UXP Registry presents a way how to reduce the need to trust to the Governing Authority. The proposed solution using the Hyperledger Fabric platform fulfills the requirements set in the Section 4.1 as follows:

- **Requirements 1 & 2:** The Governing Authority's role as controller can be main-

tained by granting explicit privileges to its and UXP Members' peer to initiate configuration changes using smart contracts.

- **Requirement 3:** Hyperledger Fabric's smart contracts allow formalizing rules that can be validated automatically.
- **Requirement 4:** Blockchain's essence that the data is stored as a hierarchical and chronologically-ordered chain of blocks with timestamp ensures that the configuration maintains its immutability and authenticity.
- **Requirement 5:** Hyperledger Fabric is a private blockchain, therefore, Governing Authority can impose regulations who can join the network (and channel) in the configuration.
- **Requirement 6:** Hyperledger Fabric provides the pluggable consensus, thus, it is more cost-effective than its alternatives. However, blockchain requires to conduct certain processes like validating blocks and maintaining the whole ledger, therefore, hardware requirements will be increased compared to the existing solution.

Implementing the blockchain-based UXP Registry answers the fundamental question, i.e., how to reduce the need to trust the correct behaviour of the Governing Authority that manages the UXP Registry. This is because blockchain helps to increase transparency in the actions carried out through UXP Registry by the following means:

- all the UXP Members store an overview of the history of configuration management by keeping the blockchain ledger;
  - therefore, the risk - that exchanging UXP messages is interrupted because global configuration cannot be received from the UXP Registry - is minimized as well;
- all the UXP Members have an overview of the configuration management principles by installing smart contracts.

4.3.1 Future work

By replacing centralized UXP Registry with a blockchain-based technology, even more perspective improvements could be implemented. For example, Hyperledger Fabric supports that there are more than one organizations who have the administrative rights, in other words, it is possible to have more than one Governing Authorities. That would allow to discover new business opportunities in context where current solution (one Governing Authority) is not suitable.

Although, federation relationship has been established to allow the interoperability of two independent UXP infrastructures, thus for example, data exchanges between two

different countries can be formed [43]. Nevertheless, implementing federation relationship can increase bureaucracy if each partner starts establishing an independent Governing Authority. The reasons behind forming the separate authority could be distrust of the partner or legal limitations of delegating the governmental tasks. Therefore, instead having two or more federated instances, but one UXP instance where certain authorities have the privileged rights could be more convenient for use cases like small island states. Even more advanced solution would be developing an entirely decentralized blockchain network where each organization would have equal rights and block validations are enforced using e.g., 2/3 majority.

Hyperledger Fabric's elements (i.e., smart contracts, channels and consortiums) should be examined further as well. Solution presented in this thesis uses only one consortium and channel where all the peers use to communicate each other, but it can be analyzed whether there are use cases where adding multiple channels could bring added value. Moreover, enforced smart contracts shall be thoroughly analyzed and developed to avoid design faults.

Supplementary registration tasks are usually handled using additional means, e.g., in Estonia management system "RIHA" is used to where additional information has to be entered to register subsystem or its certificates in X-Road. In 2014, a survey of "RIHA" [44] was carried out which identified that the most of the problems are related to the lack of automation: information is duplicated, applications are processed manually and the processing takes time. Although the thesis is focused on UXP, the results are applicable for X-Road as well. Therefore, it can be examined whether these external processes could be automated and processed within UXP instance using smart contracts.

Nevertheless, as various researches described in the Section 3.2, performance issues have to be targeted and inspected further. Implementing blockchain introduces additional processes like validating blocks and storing the ledger. For that reason, it should be additionally evaluated whether UXP Members have to act as peers for all the processes, or some process outcomings can be distinguished and transferred to another channel where only certain UXP Members have to validate the requests.

5 Summary

The goal of this thesis is to improve the existing model of UXP Registry to reduce the need to trust to the correct behavior of the Governing Authority whilst considering the advantages of new technologies. The thesis describes the blockchain-based solution for the UXP Registry that uses Hyperledger Fabric platform.

To determine how the requirements for UXP Registry influence the design of the improved UXP Registry, an analysis of the existing UXP Registry's processes is conducted. It is learned that in every instance of the UXP, there exists one central agency that is responsible for maintaining its policies. UXP Registry is a server that is used to manage these rules: it stores the security policy of the installation and distributes global configuration to the security servers of all members. It is discovered that the processes can be divided into three: individual processes that are accomplished by the Registry administrator; multi-party processes that require direct input from another party, e.g., security server administrator; and general management processes. However, that centralized management brings another challenge of UXP Registry that it acts as a trusted third party: all the members have to trust that it does not abuse its obligations.

One way to reduce the risk that the operation of the UXP instance is interfered because of the centralized management, is to replace UXP Registry as trusted third party with a distributed ledger. One of the distributed ledger technologies is blockchain which is a data structure that consists of linked records that are generated after designated time or event. Its data layer includes several attributes, e.g., Merkle tree, timestamp and asymmetric encryption. Typically in the blockchain system, there are multiple nodes that do not completely trust each other. All the nodes store an identical chain of blocks at the same time and do not rely on central authority. An important feature of blockchain is that once there is data stored into global ledger, the blockchain cannot be changed without an agreement of update validity. Furthermore, blockchain provides a smart contracts functionality that allows to reduce the risk of human error and manipulation when executing the formalized contract terms autonomously. This makes it possible to automate processes between participants without trusted third parties. These are features can be used to fulfill the requirements of UXP Registry and increase transparency in its management.

To design a new model of the UXP Registry that uses blockchain technology, various studies related to blockchain applications on the registries are analyzed and described. The analyzed studies have mostly researched how the blockchain technology is used to design an access control systems and decentralized registries. These studies can be taken into when developing new version of UXP Registry because these researches outline the achieved properties of the research subjects, for example, traceability, transparency, tamper-resistance and data integrity. The proposed architecture of UXP Registry uses private blockhain type, because the cases related to the registry operations analyzed in the thesis have implemented blockchain-based solutions using Hyperledger Fabric and Ethereum platforms. Although they both allow permissioned network and implementing smart contracts, the main difference is the used consensus mechanism. Based on that, Hyperledger Fabric is considered more feasible for implementing processes of UXP Registry and therefore chosen to implement improved UXP Registry.

The components of Hyperledger Fabric are described and explanation how they can be used in the proposed UXP Registry architecture is given. Furthermore, it is explained how the smart contracts and other features can be used to deploy the current and perspective UXP Registry processes. It is discussed which more perspective improvements could be implemented when replacing centralized UXP Registry with a blockchain-based platform. For example, it is possible to have more than one Governing Authorities which would allow to discover new business opportunities in context where current deployment is not feasible. However, it is pointed out that similarly to studies analyzed in the thesis, the performance issues have to be inspected further, because blockchain implementation requires carrying out supplementary processes.

It can be concluded that the need to trust the correct behavior of the Governing Authority can be reduced if the blockchain-based UXP Registry is implemented. Mostly because blockchain helps to increase transparency in the actions accomplished through UXP Registry.

Bibliography

[1] Z. Gao, Y. Fan, C. Wu., J. Zhang, and C. Chen, "DSES: A Blockchain-Powered Decentralized Service Eco-System," in *2018 IEEE 11th International Conference on Cloud Computing*. San Francisco: IEEE, 2018, pp. 25–32.

[2] J. Willemson and A. Ansper, "A Secure and Scalable Infrastructure for Inter-Organizational Data Exchange and eGovernment Applications," in *The Third International Conference on Availability, Reliability and Security*. Barcelona: IEEE, 2008, pp. 572–577.

[3] K. Paide, I. Pappel, H. Vainsalu, and D. Draheim, "On the Systematic Exploitation of the Estonian Data Exchange Layer X-Road for Strengthening Public-Private Partnerships," in *Proceedings of the 11th International Conference on Theory and Practice of Electronic Governance*. Galway: ACM, 2018, pp. 34–41.

[4] A. Kalja, J. Põld, T. Robal, U. Vallner, and V. Viies, "Estonian eGovernment Services: Lesson Learned," in *Proceedings of PICMET '13: Technology Management in the IT-Driven Service*. San Jose: IEEE, 2013, pp. 562–568.

[5] A. Kalja, K. Kindel, R. Kivi, and T. Robal, "eGovernment Services: How to Develop Them, How to Manage Them?" in *Proceedings of PICMET '07 : Management of Converging Technologies*. Portland: IEEE, 2007, pp. 2795–2798.

[6] A. Kalja, T. Robal, and U. Vallner., "New Generations of Estonian eGovernment Components," in *Proceedings of PICMET '15: Management of the Technology Age*. Portland: IEEE, 2015, pp. 625–631.

[7] Riigihangete Register, "Piiriüleseid teenuseid toetava X-tee arendamine," https://riigihanked.riik.ee/rhr-web/#/procurement/576227/general-info. [Accessed: 09.05.2020].

[8] Cybernetica AS, "Case Studies," https://cyber.ee/products/secure-data-exchange/case-studies/. [Accessed: 13.05.2020].

[9] Cybernetica AS, "Cybernetica launches UXP proof-of-concept with the fifth largest bank in Japan," https://cyber.ee/news/2019/05-24/, 2019. [Accessed: 13.05.2020].

[10] Nordic Institute for Interoperability Solutions, "History of NIIS," https://www.niis.org/history. [Accessed: 09.05.2020].

[11] Cybernetica AS, "Unified eXchange Platform (UXP): White paper." https://cyber.ee/products/secure-data-exchange/materials/uxp-technical-whitepaper-2020.pdf. 2020. [Accessed: 14.05.2020]

[12] R. Mändar, "UXP Portal 2.0 Functional Requirements Specification," Tartu, 2017.

[13] T. Locher, S. Obermeier, and Y. A. Pignolet, "When Can a Distributed Ledger Replace a Trusted Third Party?" in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2018.

[14] R. Zhang, R. Xue, and L. Liu, "Security and Privacy on Blockchain," in *ACM Computing Surveys*. ACM, 2019.

[15] T. T. A. Dinh, R. Liu, M. Zhang, G. Chen, B. C. Ooi, and J. Wang, "Untangling Blockchain: A Data Processing View of Blockchain Systems," in *IEEE Transactions on Knowledge and Data Engineering*. IEEE, 2018, pp. 1366 – 1385.

[16] "The Trivium method: (pertains to mind) – the elementary three," http://www.triviumeducation.com/trivium/, 2019. [Accessed: 08.06.2019].

[17] A. R. Hevner, S. T. March, J. Park, and R. S., "Design Science in Information Systems Research," in *Management Information Systems Quarterly*, 2004, pp. 75–106.

[18] Cybernetica AS, "UXP Registry Server 1.12: Installing and Configuring High Availability," 2020.

[19] Cybernetica AS, "UXP Member Management: Use Case Model," 2018.

[20] Cybernetica AS, "UXP Registry Server Management: Use Case Model," 2017.

[21] Cybernetica AS, "UXP Trust Service Management: Use Case Model," 2017.

[22] Riigi Infosüsteemi Amet, "X-tee kasutamise juhend," https://moodle.ria.ee/mod/page/view.php?id=288, 2020. [Accessed: 07.04.2020].

[23] Cybernetica AS, "UXP Service Management: Use Case Model," 2017.

[24] Cybernetica AS, "UXP Global Configuration Distribution: Use Case Model," 2018.

[25] M. Abdelhamid and G. Hassan, "Blockchain and Smart Contracts," in *ICSIE '19: Proceedings of the 2019 8th International Conference on Software and Information Engineering*. Cairo: ACM, 2019, pp. 91–95.

[26] A. Arena, P. Perazzo, and G. Dini, "Virtual Private Ledgers: Embedding Private Distributed Ledgers over a Public Blockchain by Cryptography," in *IDEAS '19: Proceedings of the 23rd International Database Applications and Engineering Symposium*. Athens: ACM, 2019, pp. 1–9.

[27] H. Du, J. Zeng, Y. An, J. Zhang, and J. Zhao, "Exploration on the Application of Blockchain in the Security System of Smart Park," in *Proceedings of the 2019 International Electronics Communication Conference*. Okinawa: ACM, 2019, p. 146–153.

[28] C. Mohan, "State of Public and Private Blockchains: Myths and Reality," in *Proceedings of the 2019 International Conference on Management of Data*. Amsterdam: ACM, 2019, p. 404–411.

[29] S. Rouhani and R. Deters, "Blockchain based access control systems: State of the art and challenges," in *2019 IEEE/WIC/ACM International Conference on Web Intelligence*. Thessaloniki: ACM, 2019, pp. 423–428.

[30] M. Samaniego, C. Espana, and R. Deters, "Access Control Management for Plant Phenotyping Using Integrated Blockchain," in *roceedings of the 2019 ACM International Symposium on Blockchain and Secure Critical Infrastructure*. Auckland: ACM, 2019, p. 39–46.

[31] O. Konashevych and M. Poblet, "Blockchain Anchoring of Public Registries: Options and Challenges," in *Proceedings of the 12th International Conference on Theory and Practice of Electronic Governance*. Melbourne: ACM, 2019, p. 317–323.

[32] N. Goderdzishvili, E. Gordadze, and N. Gagnidze, "Georgia's Blockchain-powered Property Registration: Never blocked, Always Secured - Ownership Data Kept Best!" in *Proceedings of the 11 th International Conference on Theory and Practice of Electronic Governance*. Galway: ACM, 2018, pp. 673–675.

[33] I. S. Bonomo, I. R. Barbosa, L. Monteiro, C. Bassetto, A. de Barros Barreto, V. R. P. Borges, and L. Weigang, "Development of SWIM Registry for Air Traffic Management with the Blockchain Support," in *2018 21st International Conference on Intelligent Transportation Systems*. Maui: IEEE, 2018, p. 3544–3549.

[34] A. Margheri, M. S. Ferdous, M. Yang, and V. Sassone, "A Distributed Infrastructure for Democratic Cloud Federations," in *2017 IEEE 10th International Conference on Cloud Computing*. Honolulu: IEEE, 2017, pp. 688–691.

[35] R. Almadhoun, M. Kadadha, M. Alhemeiri, M. Alshehhi, and K. Salah, "A User Authentication Scheme of IoT Devices using Blockchain-enabled Fog Nodes," in *2018*

IEEE/ACS 15th International Conference on Computer Systems and Applications. Aqaba: IEEE, 2018, pp. 1–8.

[36] A. Stanciu, "Blockchain based distributed control system for Edge Computing," in *2017 21st International Conference on Control Systems and Computer Science*. Bucharest: IEEE, 2017, pp. 667–671.

[37] M. Alblooshi, K. Salah, and Y. Alhammadi, "Blockchain-based Ownership Management for Medical IoT (MIoT) Devices," in *2018 13th International Conference on Innovations in Information Technology*. Al Ain: IEEE, 2018, pp. 151–156.

[38] "A Next-Generation Smart Contract and Decentralized Application Platform: White Paper," https://github.com/ethereum/wiki/wiki/White-Paper/. [Accessed: 13.05.2020].

[39] "Hyperledger Fabric: Open, Proven, Enterprise-grade DLT," https://www.hyperledger.org/wp-content/uploads/2020/03/hyperledger_fabric_whitepaper.pdf, 2020. [Accessed: 13.05.2020].

[40] "Hyperledger Fabric: Blockchain network," https://hyperledger-fabric.readthedocs.io/en/latest/network/network.html, 2020. [Accessed: 13.05.2020].

[41] "Hyperledger Fabric: The Ordering Service," https://hyperledger-fabric.readthedocs.io/en/latest/orderer/ordering_service.html, 2020. [Accessed: 13.05.2020].

[42] "Hyperledger Fabric: Smart Contracts and Chaincode," https://hyperledger-fabric.readthedocs.io/en/latest/smartcontract/smartcontract.html, 2020. [Accessed: 13.05.2020].

[43] R. Saarmäe, "Analysis of Configuration Management in Federated X-Road Systems," Tallinn, 2015.

[44] Riigi Infosüsteemi Amet, "RIHA kasutamise uuring 2014," https://www.ria.ee/sites/default/files/content-editors/publikatsioonid/riha_kasutamise_uuring_2014.pdf, 2014. [Accessed: 07.04.2020].