UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Software Engineering Curriculum

Kin Long Leung

A Decentralized Public Key Infrastructure for Trust Management in X-Road

Master's Thesis (30 ECTS)

Supervisor(s): Mariia Bakhtina, MA
Ahmed Awad, PhD
Raimundas Matulevičius, PhD

Tartu 2023

A Decentralized Public Key Infrastructure for Trust Management in X-Road

Abstract:

Today, Public Key Infrastructure with X.509 (PKIX) is the building block for establishing secure connections over the Internet and creating digital signatures. In PKIX, Certificate Authority (CA) is responsible for the creation of certificates and the resolution of certificate statuses. Due to the centralized architecture, CA becomes a single-point-of-failure to any network that relies on it to establish trust. By utilizing distributed ledger technology (DLT), decentralized identifiers and verifiable credentials can be verified without intermediates like CAs. They can be used to construct a Decentralized Public Key Infrastructure (DPKI) which eliminates the shortcomings of PKIX. In this thesis, we studied X-Road, a centrally managed distributed data exchange system depending on PKIX, and presented an alternate DPKI architecture that uses DLT-based decentralized identifiers and verifiable credentials to build up trust between information systems. A proof-of-concept was implemented and evaluated. The findings demonstrate that the alternative DPKI architecture enhances the trustworthiness of the data exchange system, particularly in terms of security and reliability.

Keywords:
Decentralized Public Key Infrastructure, Decentralized Identifier, Verifiable Credentials, X.509, Distributed Ledger, X-Road

CERCS: T120 Systems engineering, computer technology

Detsentraliseeritud avaliku võtme infrastruktuur usalduse haldamiseks X-Roadis

Lühikokkuvõte:

Tänapäeval on avaliku võtme infrastruktuuri kasutamine X.509 (PKIX) abil internetis turvaliste ühenduste loomise ja digitaalallkirjade loomise ehitusplokk. PKIX-is vastutavad sertifikaatide loomise ja sertifikaatide olekute lahendamise eest sertifitseerimisteenuse osutajad (CA). Keskse arhitektuuri tõttu muutub sertifitseerimisteenuse osutaja üheks ainsaks rikkepunktiks igas võrgus, mis sellele tugineb usalduse loomiseks. Hajutatud pearaamatu tehnoloogiat (DLT) kasutades saab detsentraliseeritud identifikaatorite ja kontrollitavate volikirjade abil verifitseerida usalduse loomist ilma vahendajatena toimivate sertifitseerimisteenuse osutajateta. Nende abil saab luua detsentraliseeritud avaliku võtme taristu (DPKI), mis kõrvaldab PKIX-i nõrkused. Selles artiklis uurisime X-Road'i, keskjuhitavat hajutatud andmevahetuse süsteemi, mis sõltub PKIX-ist, ning tegime ettepaneku alternatiivse DPKI arhitektuuri kohta, mis kasutab detsentraliseeritud identifikaatoreid ja kontrollitavaid volikirju, et suurendada usaldust infosüsteemide vahel. Tõestuse kontseptsioon viidi ellu ja hinnati. Leiud näitavad, et alternatiivne DPKI arhitektuur parandab andmevahetussüsteemi usaldusväärsust, eriti turvalisuse ja usaldusväärsuse osas.

Võtmesõnad:

Detsentraliseeritud avaliku võtme infrastruktuur, detsentraliseeritud identifikaator, kontrollitavad volikirjade, X.509, hajutatud pearaamat, X-Road

CERCS: T120 Süsteemitehnoloogia, arvutitehnoloogia

Contents

1 Introduction

"The Internet was built without a way to know who and what you are connecting to" [1] said Kim Cameron, a former Microsoft's Chief Architect for Identity. Public key authentication is a way for two entities to establish a trusted connection over the Internet. This requires both entities to know the public key of the other party. However, with the growing number of information systems on the Internet, it is impractical for every entity to exchange public keys in advance. Public key infrastructure using X.509 (PKIX) solves such scaling problems by introducing transitive trust, which turns trust between two parties into trust with a third [2]. In PKIX, a certificate binds a public key to an identity [3]. By introducing a trusted third party known as Certificate Authority (CA), a system can authenticate an entity during connection if the entity presents a certificate signed by a trusted CA. In turn, systems are only required to maintain a list of trusted CAs and their respective public keys for certificate verification.

While transitive trust is necessary for a trusted network to scale, PKIX has its limitations. Apart from issuing certificates, CA is responsible for delivering the certificate revocation status until the certificate expires using Certificate Revocation List (CRL) and/or Online Certificate Status Protocol (OCSP). Both CRL and OCSP suffer from scalability issues. While certificate statuses retrieved from a CRL may not always be up-to-date [4], OCSP is vulnerable to Denial-of-Service attacks and replay attacks [5]. These problems are contributed by the centralized nature of PKIX and can be mitigated by a decentralized public key infrastructure (DPKI).

Self-Sovereign Identity (SSI), an emerging peer-to-peer identity management model, brings two new standards that can facilitate a decentralized public key infrastructure [6]. First, decentralized identifier (DID) is designed to allow the identity owner to create and prove control over it without requiring permission from third-party registries [7]. Second, verifiable credential (VC) contains claims about a subject made by an issuer in a tamper-evident manner [8]. While there are different implementations of DID and VC, a decentralized public key infrastructure can be constructed by utilizing distributed ledger technology (DLT) and using DLT-based DID and DLT-based VC.

In this paper, we study X-Road, a centrally managed distributed data exchange system, and its potential limitations due to the use of PKIX for authentication and signature verification. Then, we propose an alternate architecture with DPKI built with Hyperledger Indy to reduce the shortcomings of CA and OCSP. The main research question of this paper is **[MRQ] How to establish trust between information systems using a decentralized public key infrastructure?** This is broken down into four research questions:

- [RQ1] What are the applications of decentralized public key infrastructure with decentralized identifiers and verifiable credentials?

- [RQ2] What are the requirements for X-Road with decentralized public key infrastructure?
- [RQ3] What architectural changes are necessary in X-Road to use a decentralized public key infrastructure?
- [RQ4] What is the viability of the proposed proof-of-concept implementation for X-Road with decentralized public key infrastructure?

In order to answer the research questions, this study follows the Design Science Research Method (DSRM) proposed by Hevner et al [9]. The fundamental principle of DSRM is that the knowledge and understanding of a design problem and its solution are acquired in the building and application of an artifact.

The objective of this study is to design an alternate architecture to address the limitations of PKIX and the associated business problem it brings to X-Road. A proof-of-concept (PoC) is implemented as a viable artifact.

To design an alternate architecture, we conducted a literature review to identify related work. Our design process was iterative and allowed us to refine our design choices and implementation decisions as we gained a deeper understanding of the relevant technologies and their capabilities. We followed a systematic approach throughout the study to ensure its rigor. We defined a set of design goals and use cases that require trust establishment in X-Road. From these, we elicited a list of functional requirements for the X-Road components. The PoC implementation is then evaluated based on the elicited requirements. By linking the requirements to the use cases and design goals, we ensured that all aspects of the design were considered, and the final design was aligned with the study objectives. Additionally, the changes in system quality that define trustworthiness were assessed using the assessment model proposed in preliminary work. Finally, we communicated our findings through this thesis.

Our study contributes to the application of SSI technologies in enterprise-to-enterprise use cases, particularly in the context of data exchange systems. We demonstrated that decentralized identifiers and verifiable credentials can be combined to construct a DPKI and replace X.509 certificates in PKIX. Our research also highlights the benefits and limitations of using SSI technologies in trust establishment.

The remainder of this paper is organized as follows. Section 2 provides a background on public key infrastructure, Self-Sovereign Identity and X-Road, as well as a summary of related work on onboarding and access control using Self-Sovereign Identity and distributed ledger technology. Section 3 describes the design of X-Road with DPKI and its functional requirements. In Section 4, we present the proof-of-concept implementation of X-Road with DPKI. In Section 5, we evaluate the proof-of-concept implementation against the functional requirements and the assessment model from preliminary work. In Section 6, we summarize the paper and outline our future work. This thesis was proofread using AI-based tools including Google Docs and the ChatGPT language model.

2 Background and Related Work

This section provides the background knowledge necessary to understand the subsequent work, including public key infrastructure, Self-Sovereign Identity (SSI), and X-Road. We then present research related to onboarding and access control using Self-Sovereign Identity and distributed ledger technology. In this section, we answer the research question **[RQ1] What are the applications of decentralized public key infrastructure with decentralized identifiers and verifiable credentials?**

2.1 Public Key Infrastructure

A Public Key Infrastructure (PKI) is a set of hardware, software, and policies used to distribute the binding of public keys to respective identities of entities. PKI facilitates various use cases such as establishing secure connections via SSL/TLS, encrypting documents, and creating digital signatures. In this section, we give the general background of the use of Public Key Infrastructure with X.509 (PKIX) in the Internet today and the recent development on Decentralized Public Key Infrastructure (DPKI).

2.1.1 Public Key Infrastructure with X.509

Public key infrastructure with X.509 is a widely accepted public key infrastructure used on the Internet nowadays. Since 1995, a set of standards have been developed by the Public-Key Infrastructure (X.509) Working Group, also known as the PKIX Working Group, to support X.509 based-Public Key Infrastructure. In the paper, PKIX is used to refer to X.509 based-Public Key Infrastructure following these standards [4, 5].

In PKIX, certificates are issued by Certificate Authority (CA) and there are two types of certificates: CA certificates and end-entity certificates. CA certificates are used to issue certificates to other CAs called intermediate CA or subordinate CA, and create a hierarchical system of CAs. At the top level there is a self-signed CA certificate, also known as the root CA certificate. On the other hand, end-entity certificates cannot be used to issue other certificates. Instead, end-entity certificates are used to identify an entity, such as a person, organization, or business. The list of certificates, from the root CA certificate to the end-entity certificates is sometimes known as the certificate chain. Applications such as operating systems and browsers come with a set of trusted CA certificates pre-installed. These applications can determine whether an end-entity certificate should be trusted on behalf of the users by tracing up the certificate chain and checking if it is issued by a trusted CA. To these applications, CA is known as the trusted third-party and it represents a single-point-of-failure. When the CA is compromised by attackers, the CA can issue a fraudulent certificate allowing a Man-in-the-Middle (MITM) attack which applications cannot detect [10].

Once a certificate is issued, it may need to be revoked before its expiration date under circumstances such as change of entity details, compromised CA, compromised key. PKIX defines two methods for applications to check the revocation status of a certificate. The first method is a certificate revocation list (CRL) [4]. A CRL is a time-stamped list containing revoked certificate serial numbers. A CA periodically issues a CRL to a CRL repository. Applications validating certificates should obtain a suitably recent CRL from the CRL repository and check that the certificate is not revoked. A limitation of the use of CRL is the time granularity of revocation is limited to the CRL issue frequency which could be hourly, daily, weekly. Revocation that is reported may not be reliably notified to certificate-using applications until the next CRL is issued. Also, as CRLs grow, they take up more time and resources for clients to download, process and store.

Unlike CRL, the Online Certificate Status Protocol (OCSP) allows applications to obtain the real-time revocation state of identified certificates [5]. The protocol allows a client to send a status request to an OCSP responder, who could be a CA and a CA assigned entity, and the OCSP responder returns a signed response that includes the certificate's current status. Since the response only contains information about a single certificate instead of a list of certificates, its size is often much smaller than a CRL. However, this property may lead to privacy issues as the OCSP responder may learn about the clients with which the certificate subject is interacting. Besides, the OCSP responder is vulnerable to denial-of-service attacks due to the "request/response" nature. What makes the problem worse is that the production of a cryptographic signature significantly increases response generation cycle time. When the OCSP responder becomes inaccessible, clients can only soft fail by risking to accept a revoked certificate or hard fail by stopping the operation it is trying to perform with the certificate. While precomputed responses can reduce the resource for handling queries, they allow replay attacks in which a captured good response is replayed after the certificate has been revoked.

In an effort to address the single-point-of-failure issue in PKIX and the scalability challenges of its revocation mechanisms, a decentralized public key infrastructure (DPKI) has been proposed.

2.1.2 Decentralized Public Key Infrastructure

At the first Rebooting the Web of Trust (RWOT) design workshop in 2015, an alternative approach to PKIX was introduced, called Decentralized Public Key Infrastructure (DPKI) [10]. While this was not the only use of the term DPKI, the goal of the DPKI proposed in RWOT is to ensure that no single third-party can compromise the integrity and security of the system as a whole. The way to achieve such a goal is to make use of distributed ledger technologies like blockchain. In the DPKI, an entity creates and registers a globally unique identifier, along with associated data such as public keys, in a blockchain. The entity has full control and ownership of the identifier. The blockchain is

secure through third-parties known as validators or miners. Identifiers and public keys can be obtained and verified from multiple nodes of the blockchain instead of a single trusted-third-party like the CA. Thus, the DPKI puts less burden on a single server and eliminates the MITM attack that can happen in PKIX. The concept of a globally unique identifier in DPKI has since grown into its own specification, named Decentralized Identifier (DID) [7], and has become one of the building blocks of the emerging online identity model, Self-Sovereign Identity.

2.2 Self-Sovereign Identity

Self-Sovereign Identity (SSI) is a digital identity model which focuses on user autonomy. Today, two identity models are popularized and widely used. The first is a centralized identity model where an organization identifies an user based on a shared secret, in most cases, a login password that only the organization and the user know. User data is stored and controlled by the organization. The second identity model is the federated model. This model introduces a third-party called Identity Provider (IDP) and removes users' burden of managing several login credentials using Single-Sign On (SSO). However, it has the same issue in that user data is held in and controlled by the IDP. SSI, as an emerging identity model, puts control back to users' hands with two building blocks which are Decentralized Identifier and Verifiable Credential.

2.2.1 Decentralized Identifier

A Decentralized Identifier (DID) is a globally unique identifier that does not require a centralized registration authority and is often generated and/or registered cryptographically [7]. A DID is a type of URI scheme that resolves to a DID document which contains a set of data describing the DID subject whom the DID is representing. The data includes mechanisms, such as verification methods (cryptographic public keys), that the DID subject can use to authenticate itself and prove its association with the DID.

A DID starts with a "did:" prefix, followed by a DID method and a DID method specific identifier. A DID method is defined by a specification specifying how DIDs and DID documents are created, resolved, updated, and deactivated, if these operations are supported by the method. For example, `did:sov:builder:VbPQNHsvoLZdaNU7fTBeFx` is a DID that uses the Sovrin DID method [11]. One can resolve the corresponding DID document as shown in Listing 1 by following the Sovrin DID method specification. A number of DID methods make use of distributed ledger technology (DLT) in which the data in a DID document is recorded in cryptographically signed transactions. In this paper, a DID method that makes use of DLT is referred to as DLT-based DID.

Compared to certificates in PKIX, both end-entity certificates and DID documents may contain attributes regarding the subject. However, data in DID documents is self-asserted. In that sense, a DID document is like a self-signed certificate that is not issued

```
{
  "@context": [
    "https://www.w3.org/ns/did/v1",
    "https://w3id.org/security/suites/ed25519-2018/v1",
    "https://w3id.org/security/suites/x25519-2019/v1"
  ],
  "id": "did:sov:builder:VbPQNHsvoLZdaNU7fTBeFx",
  "verificationMethod": [
    {
      "type": "Ed25519VerificationKey2018",
      "id": "did:sov:builder:VbPQNHsvoLZdaNU7fTBeFx#key-1",
      "controller": "did:sov:builder:VbPQNHsvoLZdaNU7fTBeFx",
      "publicKeyBase58": "GasrpJqMo3W9KApN4Dh62cGhYG24xhwNnCgqEKEiTKe6"
    },
    {
      "type": "X25519KeyAgreementKey2019",
      "id": "did:sov:builder:VbPQNHsvoLZdaNU7fTBeFx#key-agreement-1",
      "controller": "did:sov:builder:VbPQNHsvoLZdaNU7fTBeFx",
      "publicKeyBase58": "DUmxMDUXFTZLLdxupWFzStfdzh31XgpeoGLFL4ke8AZK"
    }
  ],
  "authentication": [
    "did:sov:builder:VbPQNHsvoLZdaNU7fTBeFx#key-1"
  ],
  "assertionMethod": [
    "did:sov:builder:VbPQNHsvoLZdaNU7fTBeFx#key-1"
  ],
  "keyAgreement": [
    "did:sov:builder:VbPQNHsvoLZdaNU7fTBeFx#key-agreement-1"
  ]
}
```


Listing 1. Example of a Resolved DID Document

by any centralized party. While an end-entity certificate can be revoked by the CA, a DID cannot be revoked due to its decentralized nature. Instead, some DID methods support removal and update of verification methods in the DID document. Also, a DID document does not convey claims about the DID subject asserted by third parties like a certificate contains claims asserted by the CA. For that, verifiable credentials are used.

2.2.2 Verifiable Credential

A verifiable credential (VC) is a tamper-proof credential encoded in such a way that authorship of it can be cryptographically verified [8]. It contains specific claims being asserted by an issuing authority about an subject. A VC can be transmitted to a verifier as a verifiable presentation (VP). A VP is a tamper-proof data that has authorship that can be trusted after a process of cryptographic verification. Like there are different DID methods, there are different VC implementations that support different features such as selective disclosure, derived predicate and revocation. While not necessary, some implementations use DIDs for expressing identifiers associated with subjects and issuers. In this paper, a VC implementation that uses DLT-based DIDs is referred to as DLT-based VC. AnonCreds is an example of DLT-based VC that supports selective disclosure, derived predicate and revocation [12].

Compared to certificates in PKIX, both end-entity certificates and verifiable credentials contain claims regarding a subject. However, verifiable credentials are usually presented to other parties in a form of verifiable presentation which may support selective disclosure and zero-knowledge proof. Also, while an end-entity certificate contains the public key controlled by the subject for verification, a verifiable presentation may not contain such information. Instead, a verifiable presentation may prove the credential-holder binding through the use of zero-knowledge proof. In such cases, a verifiable presentation cannot be used to verify a digital signature unless the verifiable credential explicitly contains the public key of the subject as a claim.

2.2.3 Agent

In SSI, an agent refers to a software program or process that performs actions on behalf of an entity, interacting with other agents [13]. A cloud agent runs on a remote server or a cloud hosting environment, as opposite to an edge agent which runs on a local device at the edge of the network. In order to perform cryptographic operations on behalf of their associated entity, an agent must have access to a secure storage, commonly known as a wallet.

2.2.4 Hyperledger Indy and Its Ecosystem

Hyperledger Indy is an open source project purpose-built for Self-Sovereign Identity[14]. It provides tools, libraries, and reusable components for providing digital identities rooted on distributed ledgers. This section explains some concepts and technologies used in the Hyperledger Indy ecosystem.

The Hyperledger Indy project includes an implementation of a distributed ledger known as the Indy ledger. The Indy ledger is a public, permissioned distributed ledger that uses the Redundant Byzantine Fault Tolerance protocol to establish a consensus between upfront well-authenticated nodes [15]. It serves as a verifiable data registry (VDR) that facilitates other SSI technologies. For examples, the Sovrin DID Method [11] and the Indy DID Method [16] utilize the Indy ledger to manage DID and DID documents. Identities created on the ledger are assigned with roles and their capabilities are governed by the permission settings of the ledger. The default settings can be found in [17]. The Indy ledger is designed such that a transaction can be endorsed [18]. If a transaction is endorsed, the transaction is signed by the endorser which could be used to evaluate trust [19].

DID communication (DIDComm) is a concept incubated within the Hyperledger Indy community [20]. The purpose of DIDcomm is to provide a secure, private communication methodology built atop the decentralized design of DIDs. When two parties communicate using DIDcomm, party A starts by looking up party B's public key by resolving party B's DID. Party A then encrypts a plaintext message using party B's public key, and adds authentication using its own private key. When party B receives the message, it decrypts the message and authenticates its origin using party A's public key. DID communication (DIDComm) is a concept incubated within the Hyperledger Indy community [20]. The purpose of DIDcomm is to provide a secure, private communication methodology built atop the decentralized design of DIDs. When two parties communicate using DIDcomm, party A starts by looking up party B's public key by resolving party B's DID. Party A then encrypts a plaintext message using party B's public key, and adds authentication using its own private key. When party B receives the message, it decrypts the message and authenticates its origin using party A's public key.

DIDcomm is designed to be transport-agnostic, meaning it can be used over various protocols such as HTTP(over TLS), WebSockets, IRC, and more. Additionally, DIDcomm is designed to be asynchronous and simplex, instead of using the duplex request-response method commonly used in web servers. Nonetheless, synchronous request-response interactions can be built on top of DIDcomm by assigning messages to the same thread. Specific interactions enabled by DIDcomm, such as connecting and maintaining relationships, issuing credentials and providing proof, are called DIDcomm protocols, also known as Aries protocols within the Hyperledger Indy community [21].

Hyperledger Aries, a project grew out of the work in Hyperledger Indy, provides interoperable tools designed for allowing trusted online peer-to-peer interactions based on

DIDs and VCs [22]. In addition to Aries protocols, another key component in the Aries project is the Aries agent. An Aries agent acts as a fiduciary on behalf of an identity owner and holds cryptographic keys that uniquely embody its delegated authorization. The agent interacts with other Aries agents using interoperable Aries protocols, facilitating secure and private communication [23].

Hyperledger AnonCreds, also known as Anonymous Credentials, is the zero-knowledge-proof-based verifiable credential implementation used in the Hyperledger Indy ecosystem [12]. The use of zero knowledge proofs offers privacy-preserving features to credential holder during the verifiable presentation process, preventing correlation with unrevealed identifiers and reducing the sharing of personally identifiable information with predicate proofs. In addition, AnonCreds provides a revocation scheme without revealing correlatable revocation identifiers.

Using AnonCreds requires a credential schema to be written in a VDR, which is typically a Indy ledger. A Credential Definition object, which references the schema as well as an issuer, must also be written to the VDR for verification. For revocable credentials, a Revocation Registry Definition containing revocation status lists must be stored and updated in the VDR.

2.3 X-Road

X-Road is an open source software that uses PKIX to facilitate data exchange between organizations [24]. An X-Road ecosystem is a community of organizations using the same instance of the X-Road software to produce and consume services. Examples of X-Road ecosystems are X-tee in Estonia and the Suomi.fi Data Exchange Layer in Finland [25].

As the owner of the X-Road ecosystem, the X-Road Operator controls which organization can join the ecosystem and defines the onboarding process and regulations that the ecosystem must follow. The X-Road Operator operates the central components of the X-Road software including the Central Server. X-Road members are organizations that have joined the ecosystem. Every X-Road member must have access to the technical component that is required for exchanging messages via X-Road, the Security Server. Figure 1 shows the overall X-Road Security Architecture.

Central Server (CS) manages and distributes the global configuration of the X-Road instance that Security Servers use for mediating the messages sent via X-Road. The global configuration consists of the registry of X-Road members and their Security Servers, as well as the security policy including a list of trusted certification authorities, a list of trusted time-stamping authorities, and configuration parameters.

Security Server (SS) handles service calls and responses between information systems of X-Road members. It encapsulates the security aspects of the infrastructure including key management, secure message exchanges, non-repudiation message creation, time-stamping, and logging. A single Security Server can host several organizations. The

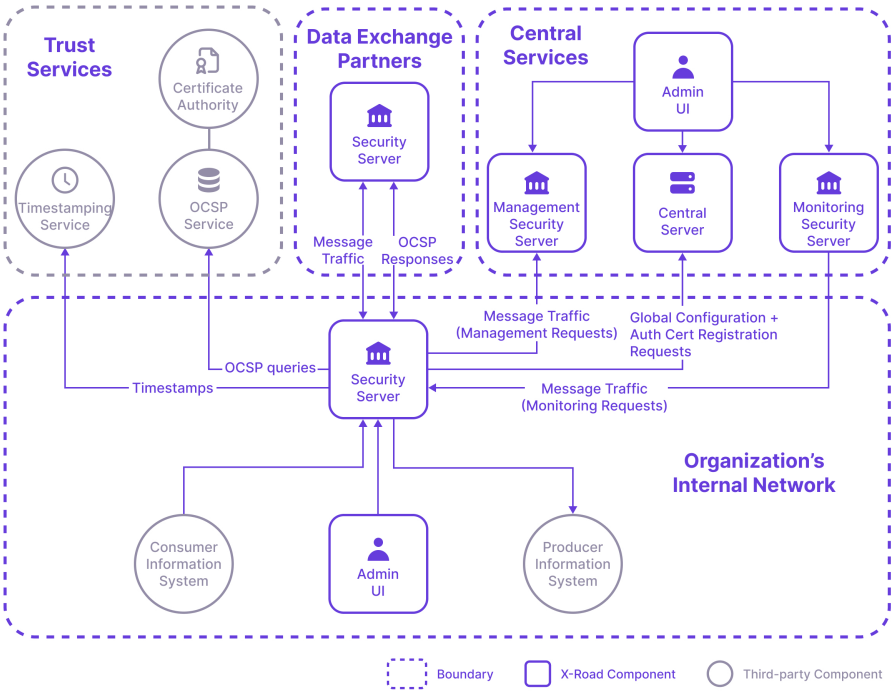

Figure 1. X-Road Security Architecture Diagram Adapted from [26]

operator of the Security Server is the server owner while the hosted organizations are the clients. The Security Server manages two types of keys: authentication key for establishing secure communication channels with other Security Servers, and signing keys for clients to sign exchanged messages. During the onboarding process, the identity of each organization and the Security Server are verified by a trusted Certification Authority and certificates are issued for the keys.

In X-Road, Security Servers use the OCSP protocol offered by the OCSP responder belonging to a trusted CA to query the certificate validity information. The Security Server does not use the nonce extension in the OCSP request so that the OCSP responses can be pre-created and cached to reduce the load in the OCSP service and increase availability [24]. However, this makes the X-Road ecosystem vulnerable to replay attacks in which a revoked certificate is accepted for authentication and signing.

A report named "Rebooting trust management in X-Road" was released in 2022 by NIIS [27]. The report discusses how the shift to the decentralized public key infrastructure can contribute to the X-Road trust model and member management. In the proposed design, X-Road members manage their DIDs and VCs in their own SSI-compatible digital wallets. DLT-based DIDs and DLT-based VCs are used to replace X.509 certificates for authentication and signing. The use of VC for verification enables privacy preserving benefits, such as selective disclosure and zero-knowledge proof, which increases the

control over the credentials the identity holder has. In our paper, we takes the proposed design from the report as a starting point and works on a proof-of-concept to demonstrate the actual use of DIDs and VCs in the X-Road for onboarding and access control.

2.4 Related Work

Soltani et al. first proposed the use of SSI and distributed ledger technology for client onboarding and Know Your Customer (KYC) [28]. They presented an onboarding framework called KYC2, which utilizes Hyperledger Indy, a public and permissioned blockchain, to improve the costly and slow KYC processes. KYC2 introduces an identity management ecosystem where clients' identity is verified once and the result is stored on their mobile devices, enabling reuse for future KYC processes. Although KYC2 has a decentralized architecture, it depends on trusted sources to perform the initial client identity proofing. Building on top of KYC2, Schlatt et al. designed a more generalized solution that can contribute to a more efficient KYC process [29]. The proposed SSI-based framework addresses the requirements of different stakeholders, including end-to-end digital processing of relevant documents, automation of manual processes, standardized exchange of KYC documents, avoiding central storage of customer data, preventing lock-in effects, and acceptance of KYC documents attested by other banks. While these two studies discuss the use of DID and VC in the client onboarding process, the clients in the discussed context are individuals who hold pairwise DIDs that are only known to the parties in connection. However, in the case of X-Road, the clients to be onboarded are public entities, so publicly readable DIDs for clients are necessary for verification.

Several studies have investigated how SSI can be utilized for access control management in the context of end-user-to-enterprise use cases. Rafael et al. proposed the Self-Sovereign Identity Based Access Control (SSIBAC), an attribute-based access control model based on decentralized identity [30]. SSIBAC maps VCs (presented as VPs) to access control policies, achieving context-based privilege and ensuring data privacy and sovereignty. Fotious et al. presented a security solution that combines VC and OAuth 2.0 to enable continuous authorization over HTTP and provide capabilities-based access control [31]. In the design, a VC verifier is employed as a policy enforcement point, intercepting HTTP requests towards a protected endpoint, and validating whether a VP can be used to execute the requested operation over a resource. This approach is more scalable and flexible than a system relying on Access Control Lists. In addition, using a DID for proof of procession in VP allows users to proactively rotate the private key without receiving a new VC, which is a good security practice. Lux et al. implemented a proof of concept decentralized OpenID Connect Provider that verifies VP and grants access to users [32]. Moreover, Lux et al. discuss using SSI technology for modifying existing digital identity standards such as X.509 certificates. They proposed two SSI-based PKIs. In transaction-based PKI, certificate attributes and DIDs are recorded

in the same transaction on a distributed ledger. In verifiable-credential-inspired PKI, certificate attributes are recorded in a verifiable credential held privately in a wallet. Fan et al. present a decentralized identity and access management, framework for IoT named DIAM-IoT which supports decentralized and user-centric data authorization [33]. It allows device owners to define user-specific rules for granting data access requests.

2.5 Answers to Research Question

In this section, we provided background information on public key infrastructure, Self-Sovereign Identity (SSI), and X-Road. We then conducted a literature review to answer the research question *"What are the applications of decentralized public key infrastructure with decentralized identifiers and verifiable credentials?"* The results indicated that DPKI with DIDs and VCs could be used for client onboarding and access control. Research on SSI is in its early stages and existing work focus on the use of SSI technology in end-user-to-enterprise use cases as well as IoT use cases. We identified the lack of discussion about modeling decentralized public key infrastructure with SSI for entreprise-to-enterprise use cases. Therefore, we are motivated to base our contribution on the subject and suggest an alternate architecture for X-Road, a data exchange system that relies on PKIX, to migrate to a DPKI constructed with DIDs and VCs.

3 Design of X-Road with DPKI

The section presents the design goals of X-Road with DPKI, as well as the use cases and elicited requirements. In this section, we answer the research question **[RQ2] What are the requirements for X-Road with decentralized public key infrastructure?** It is broken down into three sub-research questions:

- [RQ2.1] What are the design goals for X-Road with decentralized public key infrastructure?
- [RQ2.2] What are the use cases for X-Road with decentralized public key infrastructure?
- [RQ2.3] What are the functional requirements for X-Road with decentralized public key infrastructure?

3.1 Assumptions

The design of X-Road with DPKI assumes that enterprises adopt the Self-Sovereign Identity model, in which they control their own SSI agents in the cloud. Organizations' SSI agents have to hold verifiable credential(s) required by the X-Road governing authority in order to join an X-Road instance. An organization is considered onboarded once its DID and service endpoint can be publicly accessed and verified. In the design, this is achieved by recording the organization's DID and service endpoint on the distributed ledger, and the transaction is endorsed by the X-Road governing authority.

3.2 Design Goal

The design goals of X-Road with DPKI includes existing design goals of X-Road [24] with the following new design goals added:

G1: Decentralized In the existing X-Road with PKIX, the data exchange happens between organizations only after two parties have established a secure connection with the involvement of CA or OCSP responder. The design in this study aims to support secure connection establishment without intermediaries.

G2: Granular Access Control In the existing X-Road with PKIX, discretionary access control is implemented to restrict access to service based on the identity of the subject (a subsystem or the access rights group to which the subsystem belongs) [34]. A design goal of this study is to enable a more granular attribute-based access control by replacing X.509 certificates with verifiable credentials that support selective disclosure and zero-knowledge proof.

G3: Automated Member Onboarding In addition to the existing design goals, the design also aims to demonstrate how member onboarding processes can be sped up with automated verification of credential presentations without reliance on any third parties.

3.3 Use Cases

This section describes the use cases of the X-Road Central Server and Security Server, which have an impact on how trust is established between two parties using VCs. Each use case is defined to satisfy one or more design goals.

3.3.1 Use Cases of Central Server

Figure 2 depicts the use cases of X-Road Central Server and the actors of the use cases. Table 1 presents *Use Case 1: Configure and Start Central Server*, where the Central Server administrator configures and initiates the Central Server. It is assumed that the Central Server has been installed by the administrator beforehand. The main success scenario involves the administrator configuring the SSI agent interfaces, the verifiable presentation request for data exchange, and the validity time of a verifiable presentation, followed by starting the Central Server. Upon successful completion of the use case, the Central Server should be ready to provide onboarding service. The use case aligns with the design goal G2, which aims to provide granular access control. Table 2 presents *Use Case 2: Automated Member Onboarding*, in which the Security Server uses the system to automatically onboard member, i.e. to get an endorsement for an organization DID registration transaction.

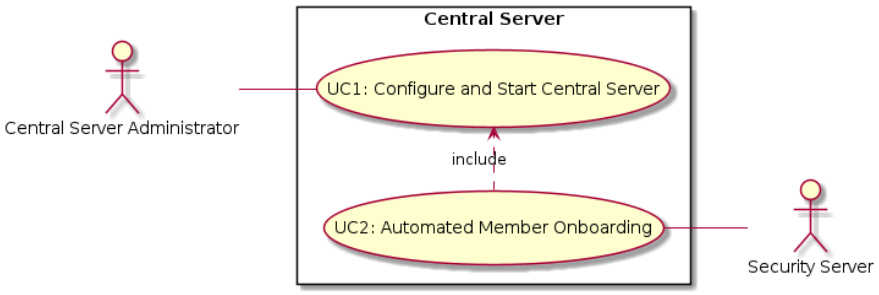

Figure 2. Use Case Diagram for X-Road Central Server

Table 1. Use Case 1: Configure and Start Central Server

ID:	UC1
Name:	Configure and Start Central Server
Description:	CS administrator configures and starts the Central Server.
Goals:	G2
System:	Central Server
Level:	User
Pre-conditions:	Central Server has been installed by the administrator.
Post-conditions:	Central Server is ready to provide onboarding service.
Triggers:	-
Main success scenario: 1. CS administrator configures the SSI agent interfaces, VP request for data exchange and the time of VP validity. 2. CS administrator starts the Central Server.	

3.3.2 Use Cases of Security Server

Figure 3 depicts the use cases of X-Road Security Server and the actors of the use cases. *Use Case 3: Configure and Start Security Server* is described in Table 3, which outlines how the Security Server administrator configures the communication interfaces between the Security Server and an SSI agent, as well as the verifiable presentation request for access control and the validity time of a verifiable presentation. Table 4 presents *Use Case 4: Send Onboarding Request*, where the Security Server administrator uses the Security Server user interfaces to send an onboarding request. Table 5 and Table 6 describe *Use Case 5: Consume Service* and *Use Case 6: Provide Service*, respectively. In these use cases, information systems of service consumers and service providers exchange data through their own Security Servers, which perform access control, message signing, and verification.

Table 2. Use Case 2: Automated Member Onboarding

ID:	UC2
Name:	Automated Member Onboarding
Description:	Central Server automatically verifies and endorses the DID registration transaction request from Security Server.
Goals:	G1, G3
System:	Central Server
Level:	System
Actor:	Security Server
Pre-conditions:	Central Server is configured and started.
Post-conditions:	-
Triggers:	-

Main success scenario:

1. Security Server sends a DID registration transaction endorsement request to the Central Server.
2. Security Server receives a VP request for onboarding from the Central Server.
3. Security Server sends a VP to the Central Server.
4. Central Server verifies the received VP from the Security Server.
5. Security Server receives the endorsed DID registration transaction from the Central Server.

Extensions:

4.a Central Server receives an invalid VP from the Security Server:

1. Central Server refuses to endorse the DID registration transaction.

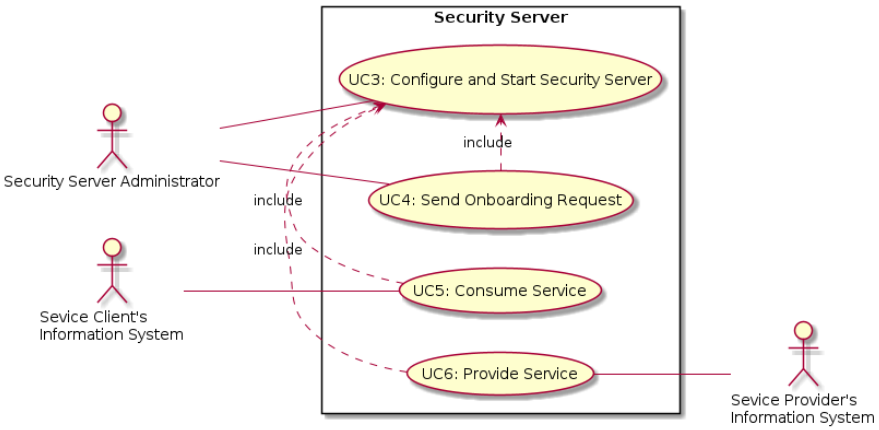

Figure 3. Use Case Diagram for X-Road Security Server

Table 3. Use Case 3: Configure and Start Security Server

ID:	UC3
Name:	Configure and Start Security Server
Description:	SS administrator configures and starts the security server.
Goals:	G2
System:	Security server
Level:	User
Actor:	SS administrator - Security server administrator
Pre-conditions:	The security server has been installed by the administrator.
Post-conditions:	The security server is ready for sending onboarding request.
Triggers:	-
Main success scenario: 1. SS administrator configures the SSI agent interfaces, VP request for data exchange and the time of VP validity. 2. SS administrator starts the security server.	

Table 4. Use Case 4: Send Onboarding Request

ID:	UC4
Name:	Send Onboarding Request
Description:	Security Server administrator sends an onboarding request.
Goals:	G1, G3
System:	Security Server
Level:	User
Actor:	SS administrator - Security Server administrator
Pre-conditions:	The organization's SSI agent contains VC(s) required to create VP for onboarding.
Post-conditions:	The organization's DID registration transaction is endorsed by the Central Server and written on the distributed ledger. The Security Server is ready for exchanging X-Road messages.
Triggers:	-

Main success scenario:

1. SS administrator selects to initiate the DID registration of the organization in the Security Server UI.
2. Security Server creates a DID and a DID registration transaction.
3. Security Server sends the DID registration transaction to the Central Server for endorsement.
4. Security Server receives a VP request for onboarding from the Central Server.
5. Security Server creates and sends a VP to the Central Server.
6. Security Server receives the DID registration transaction with the Central Server's endorsement.
7. Security Server writes the endorsed DID registration transaction to the distributed ledger.
8. SS administrator sees a successful message in the Security Server UI.

Table 5. Use Case 5: Consume Service

ID:	UC5
Name:	Consume Service
Description:	An X-Road member's information system consumes service provided by another X-Road member's information system.
Goals:	G1, G2
System:	Security Server
Level:	System
Actor:	Service client's IS - Service client's information system
Pre-conditions:	X-Road member is onboarded.
Post-conditions:	The messages exchanged are signed and logged.
Triggers:	-

Main success scenario:

1. Service client's IS sends a request message targeted at a service provider to the Security Server.
2. Security Server exchanges VPs with the service provider's Security Server.
3. Security Server signs the request message.
4. Security Server transmits the request message to the service provider's Security Server.
5. Security Server receives a response message from the service provider's Security Server.
6. Security Server verifies the signature on the response message.
7. Security Server forwards the response message to the service client's IS.
8. Service client's IS receives the response message from the Security Server.

Extensions:

2.a Service provider's Security Servers fails to construct a valid VP:

   1. Security Server returns an error message back to the information system.

6.a The signature on the response message is invalid:

   1. Security Server returns an error message back to the information system.

Table 6. Use Case 6: Provide Service

ID:	UC6
Name:	Provide Service
Description:	An X-Road member's information system provides service to another X-Road member's information system.
Goals:	G1, G2
System:	Security Server
Level:	System
Actor:	Service provider's IS - Service provider's information system
Pre-conditions:	X-Road member is onboarded.
Post-conditions:	The messages exchanged are signed and logged.
Triggers:	Service provider's IS receives request message from Security Server.
Main success scenario: 1. Service provider's IS sends a response message to the Security Server. 2. Security Server signs the response message. 3. Security Server transmits the response message to the service client's Security Server.	

3.4 Functional Requirements

Based on the use cases described, the functional requirements of X-Road components in the design are listed. Corresponding acceptance criteria are described for every requirement. Each requirement is defined to fulfill one or more use cases. Table 7 lists the common functional requirements for both Central Server and Security Server. For example, the first functional requirement (ID: 1) specifies that a component must present a verifiable credential with a cryptographic proof of possession to create a verifiable presentation in order to fulfil UC2, UC4, UC5, UC6. The requirement is considered to be met if a verifiable presentation created by the component can be cryptographically verified that it is created by the holder of verifiable credential. Table 8 lists the functional requirements for Central Server, while Table 9 lists the functional requirements for Security Server.

Table 7. Common Functional Requirements of X-Road Components

ID	***Use cases***	***Functional Requirement***	***Acceptance Criteria***
1	UC2, UC4, UC5, UC6	A component must present a VC with a cryptographic proof of possession to create a VP.	A VP created by a component can be cryptographically verified that it is created by the holder of VC.
2	UC2, UC4, UC5, UC6	A component must verify the holder binding of every received VP.	A component cryptographically verifies that every received VP is issued to the prover.
3	UC2, UC4, UC5, UC6	A component must verify the authenticity of every received VC without reliance on a single trusted-third party.	A component cryptographically verifies that every received VC matches the properties in the presentation request using information obtained from the distributed ledger.
4	UC2, UC4, UC5, UC6	A component must verify the revocation status of every received VC without reliance on a single trusted-third party.	A component cryptographically verifies the revocation status of every received VC. The revocation status can be obtained from the distributed ledger.
5	UC2, UC4, UC5, UC6	A component must request VP using protocol that prevents replay attack.	A component cryptographically verifies a VP that is uniquely bound to the presentation request.

Table 8. Functional Requirements of Central Server

ID	***Use cases***	***Functional Requirement***	***Acceptance Criteria***
6	UC1	Central Server must allow Central Server administrators to customize the attributes in the VP request for onboarding purposes.	Central Server administrators can change the attributes in the VP request for onboarding purposes.
7	UC2	Central Server must onboard qualified organizations by assisting them to register their DID on the distributed ledger.	Central Server endorses DID registration transactions submitted by a Security Server which has submitted a valid VP. Central Server does not endorse DID registration transactions submitted by a Security Server which has not submitted a valid VP.

Table 9. Functional Requirements of Security Server

ID	*Use cases*	*Functional Requirement*	*Acceptance Criteria*
9	UC3	Security Server must allow Security Server administrators to customize the attributes in the VP request for access control.	Security server administrators can set the attributes in the VP request for access control before starting the server
10	UC4	Security Server must allow its clients to send requests to the Central Server to write their DIDs on the distributed ledger.	Security Server clients can send DID registration transaction endorsement requests to the Central Server using an API provided by the Security Server.
11	UC5	Security Server must allow its clients to send messages to service providers in the same X-Road network.	Security Server clients can send messages to service providers in the same X-Road network using an API provided by the Security Server.
12	UC5, UC6	Security Server must send a VP request for access control to the other member's Security Server after receiving a message from its client targeted to another member if it has no record of valid VPs sent by the other member's Security Server.	Security Server sends a VP request for access control to the other member's Security Server after receiving a message from its client targeted to another member and it has no record of valid VPs sent by the other member's Security Server.
13	UC5, UC6	Security Server must transmit a message from its client to another Security Server only if a VP submitted by the other Security Server is valid.	Security Server transmits messages from its client to another Security Server which has submitted a valid VP. Security Server does not transmit messages from its client to another Security Server which has not submitted a valid VP.
14	UC5, UC6	Security Server must transmit a signature computed from the message sent from its client and a private key associated with its client's DID along with the original message to another Security Server.	The signature sent along with a message can be verified that it was computed from the message and a private key associated with the sender's DID.
15	UC5, UC6	Security Server must transmit a message from another Security Server to its client only if the message signature is valid.	Security Server transmits messages from another Security Server to its client if the message signature is valid. Security Server does not transmit messages from another Security Server to its client if the message signature is invalid.

3.5 Answers to Research Questions

In this section, we discussed the design goal, use cases and the functional requirements of X-Road with DPKI to answer the research question *[RQ2] What are the requirements for X-Road with decentralized public key infrastructure?* It is broken down into three sub-research questions and we gave answers to each of them.

[RQ2.1] What are the design goals for X-Road with decentralized public key infrastructure? The design goals of X-Road with DPKI should align with current implementation of X-Road with improvement on removing third parties to secure established connection, more granular access control and automated onboarding process to justify the migration.

[RQ2.2] What are the use cases for X-Road with decentralized public key infrastructure? To achieve the design goals, six use cases have been identified. Each use case describes the expected behaviour of the X-Road component in context. They cover X-Road server configuration for SSI agent integration and verifiable presentation requests, as well as automated member onboarding and data exchange with DPKI.

[RQ2.3] What are the functional requirements for X-Road with decentralized public key infrastructure? The functional requirements for both X-Road Central Server and Security Server are identified to fulfill the use cases. Each requirement has corresponding acceptance criteria for evaluation.

4 Implementation of X-Road with DPKI

In the previous section, we established the requirements for implementing a decentralized public key infrastructure for X-Road. In this section, we discuss our implementation of proof-of-concept (PoC) of X-Road using decentralized public key infrastructure. We introduce the PoC implementation that integrates with Hyperledger Indy ledger and open source cloud agent framework Hyperledger Aries Cloud Agent Python. We describe how the X-Road components are replaced by external components and how should their interfaces be modified. Finally, we discuss how two network activities, namely member onboarding and data exchange, that require trust establishment are performed in the PoC.

Through the implementation of the PoC, we answer the research question **[RQ3] What architectural changes are necessary in X-Road to use a decentralized public key infrastructure?** This research question is broken down into three guiding questions:

- [RQ3.1] What components are no longer needed in X-Road using decentralized public key infrastructure?
- [RQ3.2] What components should be changed in X-Road using decentralized public key infrastructure?
- [RQ3.3] How trusted activities are performed in X-Road using decentralized public key infrastructure?

4.1 Design Decisions

To utilize a decentralized public key infrastructure with SSI technologies, X-Road has to select:

- one or more DID method(s)
- one or more implementation(s) of verifiable credential
- one or more SSI agent(s) for integration

These selections should be based on the requirements identified in the previous section. Specifically, the DID method should store the data to produce DID documents on a distributed ledger. The implementation of verifiable credential should support revocation status check using revocation registry published on a distributed ledger. The same distributed ledger can be used for the DID method and the implementation of verifiable credential. Lastly, X-Road Central Server and Security Server should support integration with an SSI agent that supports the selected DID method and verifiable credential implementation.

The integration with the SSI agent can be one of the following two ways depending on the interfaces of the selected SSI agent. If the SSI agent implements its own connection protocol that is required for credential presentation, the SSI agent should be placed between an X-Road server (Central Server or Security Server) and the external network, as shown in Figure 4. In other words, two X-Road servers should communicate via their own SSI agents. Alternately, if the SSI agent does not implements its own connection protocol and is simply a software to store private data such as keys and VCs, an X-Road server should connect with the external network directly. That way, the SSI agent becomes an external service that it consumes, as shown in Figure 5, and two X-Road servers should communicate directly.

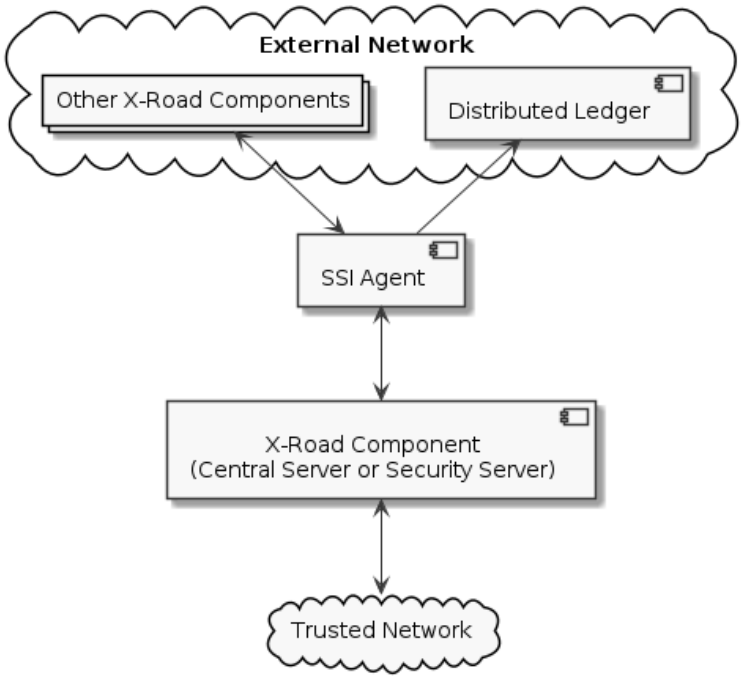

Figure 4. SSI Agent Integration Option 1

4.2 Proof-of-Concept Implementation

In the PoC, Sovrin DID method is used [11]. DID data is stored on an instance of Hyperledger Indy, which is a distributed ledger purpose-built for Self-Sovereign Identity and verifiable credentials [14]. AnonCreds is chosen to be the implementation of verifiable credential since AnonCreds supports revocation status check using revocation registry published on Hyperledger Indy [12]. X-Road components are integrated with Aries Cloud Agent Python which is an open source SSI agent that supports Sovrin

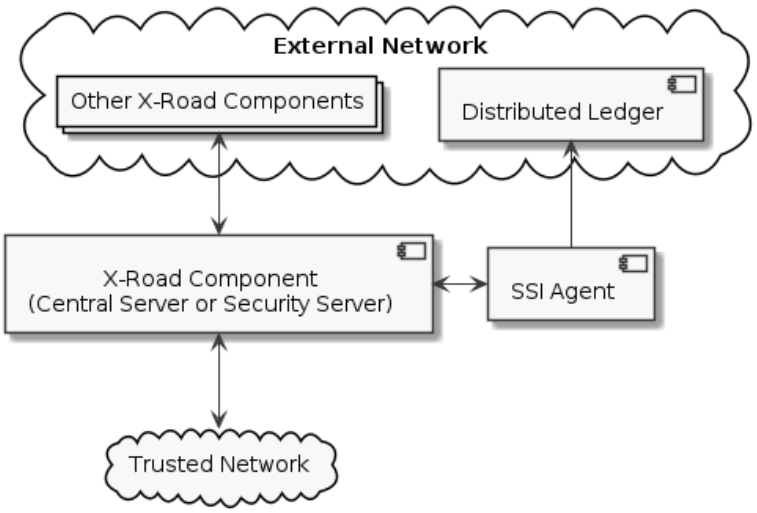

Figure 5. SSI Agent Integration Option 2

DID method and AnonCreds [35]. These open-source technologies are used in production by BC Digital Trust from the Government of British Columbia [36]. The source code of the PoC is available at `https://gitlab.cs.ut.ee/ssi-xroad/ssi-xroad` and the source code of the modified Aries Cloud Agent Python is available at `https://gitlab.cs.ut.ee/ssi-xroad/aries-cloudagent-python`.

4.2.1 Architecture

In a world where the Self-Sovereign Identity model is adopted and organizations control their own SSI agents, a number of components in existing X-Road can be externalized and delegated to external components such as SSI agents and the distributed ledger. This subsection discusses the setting of these external components, as well as how X-Road components could evolve to work with external components to perform its functionalities. Figure 6 shows the security architecture of the PoC, which can be compared with the original security architecture shown in Figure 1.

Hyperledger Indy (Distributed Ledger) Hyperledger Indy is distributed ledger purpose-built to be a verifiable data registry for storing DIDs and DID documents, credential schema and revocation status [14]. In the PoC, VON Network which is an open source project maintained by the Government of British Columbia is used to set up a Hyperledger Indy Node network for development [37]. It comes with development features that help the demonstration of the PoC including UI for browsing ledger transactions and status of nodes. There are 4 validator nodes running in the VON network.

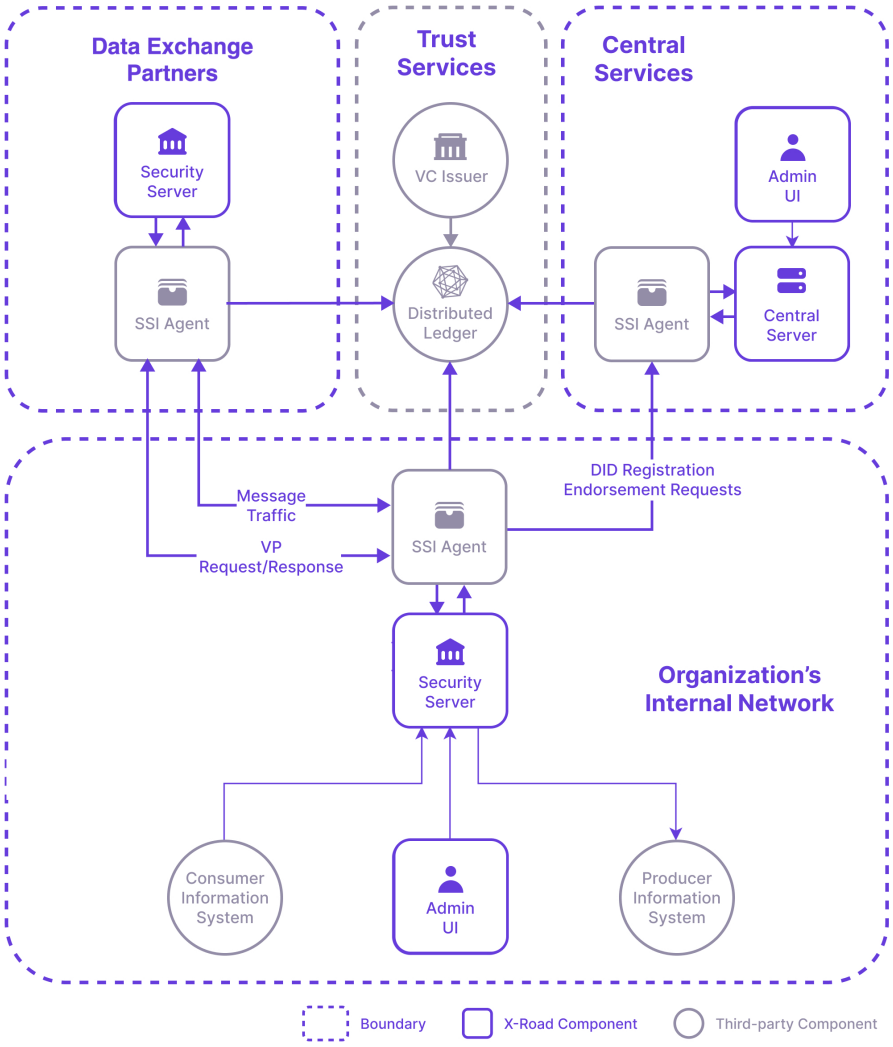

Figure 6. PoC Security Architecture Diagram

The authorization rules of the network are kept as default. With the default settings, a user with role TRUSTEE/STEWARD/ENDORSER can add a new identity owner(user without role). This rule affects the onboarding process in PoC which is to have an endorser endorse a DID registration transaction from an organization. The configuration of the Indy network is out of scope of this PoC. In a production network like the Sovrin MainNet, DID registration operation should be permissioned. There should be more validator nodes in the network. The authorization rules should also be carefully reviewed since it affects the trust model of the network as well as the network consumers. For example, the rule mentioned above can be set that a trustee or steward can add a new identity owner but at least 3 endorsers are required to add a new identity owner.

Aries Cloud Agent Python (SSI agent) In the PoC, it is assumed that each organization controls an Aries Cloud Agent Python (ACA-Py) instance as their SSI agent. The PoC uses a number of Aries protocols supported by ACA-Py. DID Exchange Protocol is used for making connections between agents [38]. Issue Credential Protocol 2.0 is used for issuing credentials [39]. Present Proof Protocol 2.0 is used for presenting proofs [40].

Since one of the design goals of X-Road is to have all messages to be usable as digital evidence, creating non-repudiable messages (signed message) is one of the features in X-Road. SSI agent that controls the identity verification key should be responsible for creating signatures without sharing the private key with other components. However, ACA-Py does not expose an interface for signing payload. Instead, the signing function is used internally within protocol implementations. At the time of writing, the only protocol that supports non-repudiable message exchange is Non-Repudiable Signature for Cryptographic Envelope but it is in ”Proposed” state and is not supported by ACA-Py [41]. Therefore, part of the PoC is the implementation of a simple non-repudiable message exchange protocol that supports the PoC use cases.

In the non-repudiable message exchange protocol, there are two roles, ”Requester” and ”Responder”, and three message types, ”Request”, ”Response” and ”Denial”. Requester initiates the protocol by making a request to a responder and the responder can either deny or send a response. The payload within a request or response message is signed by the sender verification key. In the future this protocol is expected to be replaced by a well-recognized community-accepted protocol.

In the PoC, ACA-Py replaces the Signer component in the existing X-Road Security Server which is responsible for managing keys and certificates used for signing messages. ACA-Py is also responsible for transmitting messages in a secure channel (with DIDcomm) which removes part of the responsibility from the Proxy component in Security Server.

X-Road Central Server The Central Server in PoC is a Spring Boot Application developed with Java. It provides automated endorsement service to Security Servers

connected with it after requesting and receiving a verifiable presentation. Figure 7 shows the components and interfaces of the Central Server.

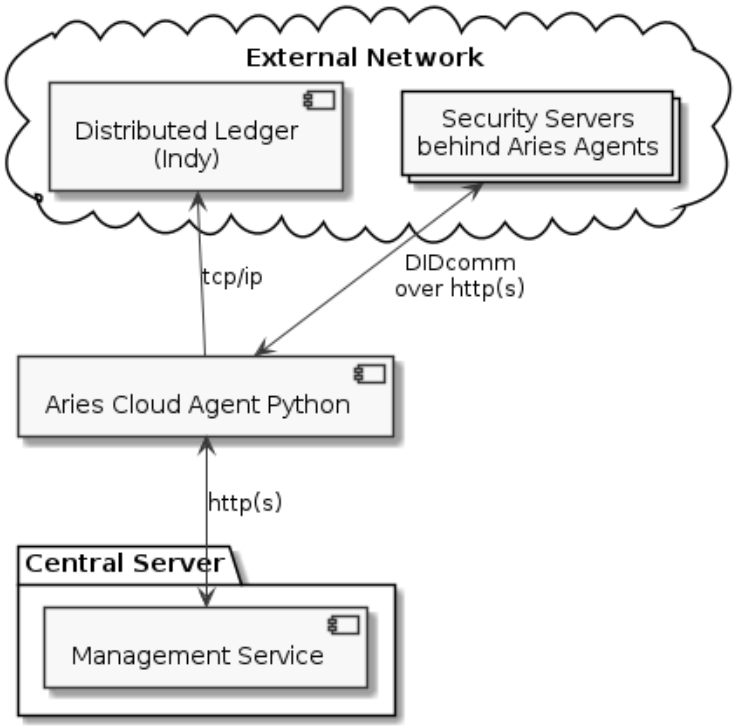

Figure 7. Components and Interfaces of Central Server in PoC

The proof request Central Server sends to Security Servers for onboarding is configurable. In the demonstration, the proof request requests three attributes ”name”, ”registryCode” and ”established” from a credential with schema name ”Business Registration Certificate”. Listing 2 shows an example of onboarding proof request specified in JSON format.

X-Road Security Server As some core functionalities of Security Server are delegated to ACA-Py, a lightweight Security Server is implemented for handling the operational logic of X-Road in the PoC, including member onboarding, message mediation and access control. While existing Security Server supports multi-tenancy, the Security Server in PoC is designed for a single organization. Multi-tenancy can be achieved by using the multi-tenancy feature of ACA-Py. The Security Server in PoC is a Spring Boot Application developed with Java. Figure 8 shows the components and interfaces of the PoC Security Server.

The Security Server Management REST API exposes an endpoint for member onboarding. When invoked, the Security Server instructs ACA-Py to connect with that

```
{
  "name": "Proof of Business Registration Certificate",
  "version": "1.0",
  "requested_attributes": {
    "0_business_uuid": {
      "names": ["name", "registryCode", "established"],
      "restrictions": [{
        "schema_name": "Business Registration Certificate"
        }]
    }
  },
  "requested_predicates": {}
}
```

Listing 2. Onboarding Proof Request

Central Server's Aries agent and presents proof of possession for the required credentials for onboarding. Listing 3 shows an example onboarding request.

```
POST /management/onboard HTTP/1.1
Host: localhost:8001
Accept: application/json
Content-Type: application/json
Content-Length: 0
```

Listing 3. Onboarding Request

The Security Server Proxy API is responsible for mediating HTTP messages between service consumers and service providers. It exposes an endpoint (/x-road/{information system's DID}/{path}) for forwarding requests to other X-Road members' information systems. The Proxy encodes HTTP requests from service consumers and decodes HTTP responses from service providers and sends/receives the payload through ACA-Py using the non-repudiable message exchange protocol. Listing 4 shows a HTTP request sent to an information system identified by `did:sov:2wJPyULfLLnYTEFYzByfUR` through a Security Server.

Before forwarding requests (as a Security Server of the service consumer) and after receiving requests (as a Security Server of service provider), the Proxy sends a verifiable

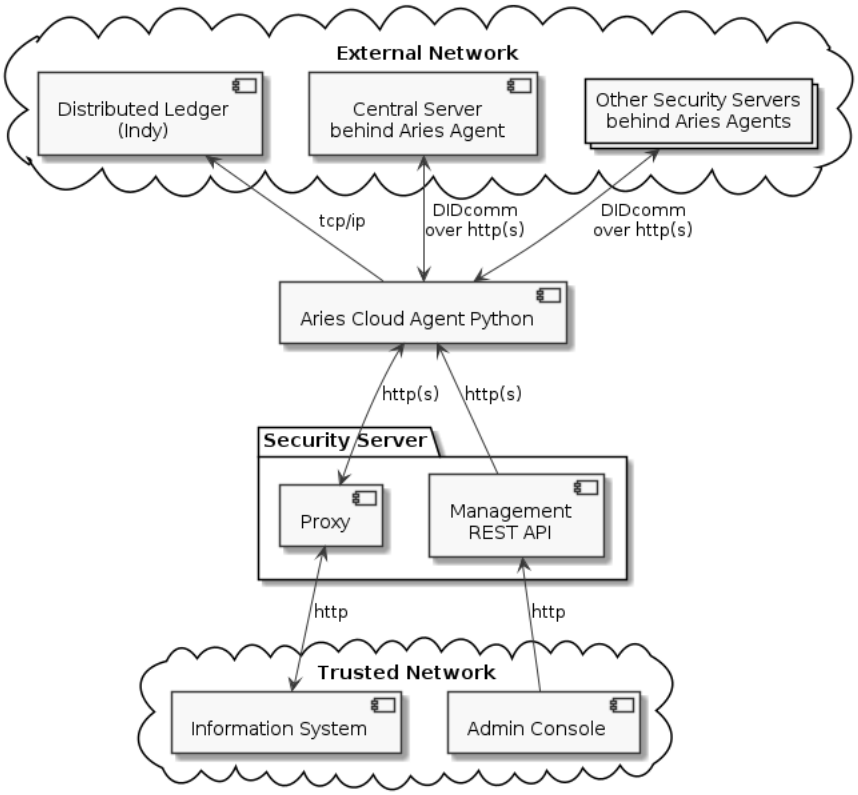

Figure 8. Components and Interfaces of Security Server in PoC

```
GET /x-road/2wJPyULfLLnYTEFYzByfUR/books/3 HTTP/1.1
Host: localhost:8001
```

Listing 4. HTTP Request through Security Server

presentation proof request to the other party. If any party fails to present the requested proof, the message exchange process aborts and an error message is returned to the requester. The default proof request is the same as the one for onboarding (see Listing 2) but it can be configured to be more fine-grained, for example, requesting more than one credential and include predicates (see Listing 5).

A PoC Security Server UI is developed for member onboarding and viewing message exchange records. For demonstration purposes, it also provides UI for viewing connection and credential records in ACA-Py and requesting credentials from Issuer.

```
{
  "name":
    "Proof of BR And Startup Membership",
  "version": "1.0",
  "requested_attributes": {
    "0_business_uuid": {
      "names": ["name", "registryCode", "established"],
      "restrictions": [{
            "schema_name": "Business Registration Certificate"
        }]
    },
    "0_startup_uuid": {
      "names": ["name", "membershipCode"],
      "restrictions": [{
            "schema_name": "Startup Estonia Membership"
        }]
    }
  },
  "requested_predicates": {
    "0_startDate_GE_uuid": {
      "name": "startDate",
      "p_type": ">=",
      "p_value": 20200101,
      "restrictions": [{
        "schema_name": "Startup Estonia Membership"
      }]
    }
  }
}
```

Listing 5. Custom Proof Request for Access Control

4.2.2 Network Activities

This subsection discusses how two core network activities on X-Road is performed in the PoC. The first activity discussed is member onboarding, in which the X-Road Governing Authory helps to register the organziation's DID on the ledger, after requesting and verifying a verifiable presentation submitted by the organization. The second activity is message exchange and access control, in which two X-Road members exchange verifiable presentations to verify each other's identities and access rights.

Member Onboarding Figure 9 is a sequence diagram that illustrates the success scenario of member onboarding. In the success scenario, the applicant holds the required verifiable credentials in its wallet and can present proof for onboarding.

Applicants wishing to join the PoC X-Road instance initiates the onboarding flow using the Management REST API. Information about the Central Server (endorser) for X-Road onboarding should be made publicly available on the Indy ledger. In the PoC, the Central Server's DID can be found using VON network's transaction browser. Upon request, the applicant's Security Server connects with the Central Server through their own Aries agents using the DID exchange protocol. Once connected, both Security Servers set their roles for the connection. Applicant's Security Server sets its role to be "Transaction Author", whereas the Central Server sets its role to be "Transaction Endorser". The Central Server then requests proof of possession for the required credentials for onboarding using Aries Present Proof Protocol 2.0. Once the proof presentation is completed, the applicant's Security Server creates a new DID and sends a DID registration transaction to the Central Server for endorsement. The Central Server verifies that it has received the required presentation proof from applicant and endorses the transaction (by adding its signature). Finally, the transaction is sent back to the applicant's Security Server who then requests network nodes to write the transaction to the ledger. The onboarding process completes when the applicant's DID is registered on the ledger so that it can be discovered by other X-Road members.

Message Exchange and Access Control Figure 10 is a sequence diagram that illustrates the success scenario of a message exchange. In the success scenario, both parties are able to present requested proofs for access control. Service client initiates a message exchange flow by sending a request to the proxying endpoint of its Security Server. Service client's Security Server connects with service provider's Security Server. Once connected, the service consumer's Security Server sends a presentation proof request to the service provider's Security Server for authentication and authorization. After receiving and verifying the proof, the service consumer's Security Server encodes the HTTP request as a JSON payload and sends it through ACA-Py using the non-repudiable message exchange protocol. Upon request, the service provider's Security Server sends a presentation proof request to the service consumer's Security Server for authentication

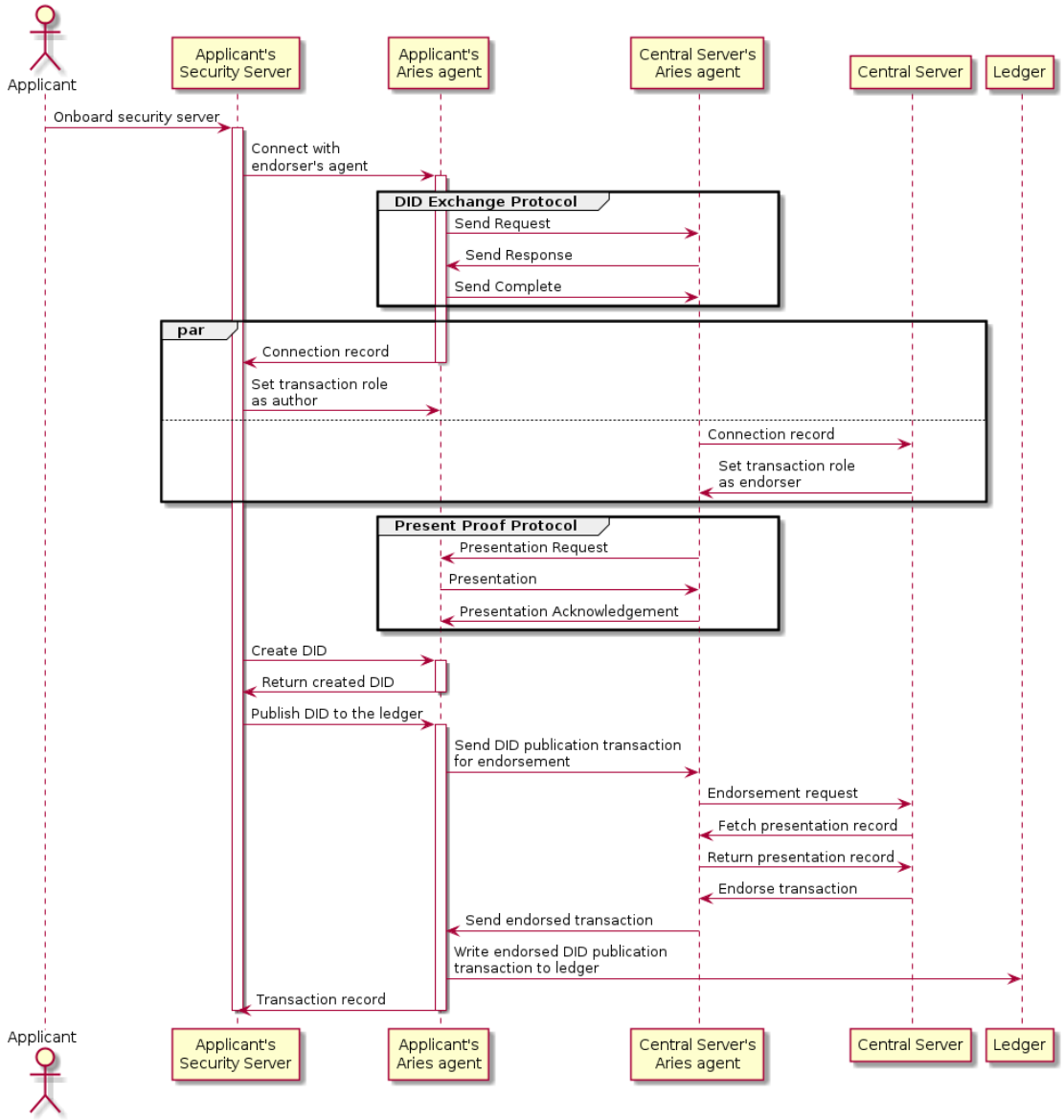

Figure 9. Member Onboarding in PoC

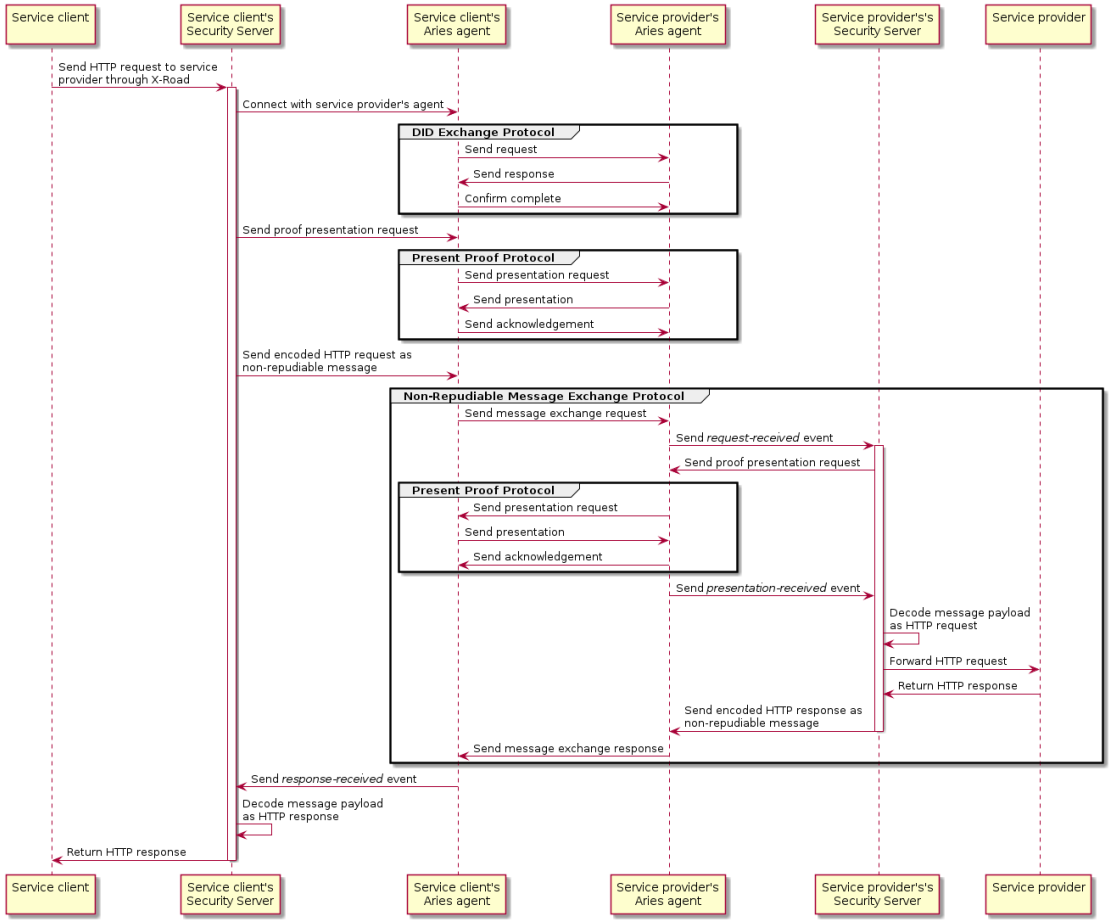

Figure 10. Message Exchange in PoC

and authorization. After receiving and verifying the proof, the service provider's Security Server decodes the message payload and forwards the HTTP request to its information system. The information system returns a HTTP response which is then encoded and sent back to the service consumer through ACA-Py using the non-repudiable message exchange protocol. If any party fails to present the requested proof, the message exchange protocol aborts and an error message is returned to the service consumer.

4.3 Answers to Research Questions

In this section, we presented our implementation of proof-of-concept (PoC) of X-Road using decentralized public key infrastructure. The research question *[RQ3] What architectural changes are necessary in X-Road to use a decentralized public key infrastructure?* is broken down into three sub-research questions and we gave answers to each of them.

[RQ3.1] What components are no longer needed in X-Road using decentralized public key infrastructure? The Signer component in the existing X-Road Security Server is no longer needed in X-Road using decentralized public key infrastructure. The Signer component is currently responsible for managing keys and certificates used for signing messages and it is expected to be replaced by the secure storage implementation in SSI agent. SSI agent is also responsible for transmitting messages in a secure channel using DIDcomm which removes part of the responsibility from the Proxy component in Security Server.

[RQ3.2] What components should be changed in X-Road using decentralized public key infrastructure? Both Security Server and Central Server need to integrate with an SSI agent which includes consuming API provided by the SSI agent and listening events emitted from the SSI agent. The Proxy component in Security Server should transform HTTP messages into payloads that are accepted by the SSI agent, and vice versa.

[RQ3.3] How trusted activities are performed in X-Road using decentralized public key infrastructure? Secure connection and credential exchange protocols built atop DID are required to perform trusted activities in X-Road including member onboarding and data exchange. For example, Aries protocols can be used through Aries agent to perform these activities.

5 Evaluation

In this section, we evaluate the PoC implementation against the functional requirements identified in Section 3. To achieve that, we set up an X-Road network instance with our PoC components together with other auxiliary components for evaluation. We simulate member onboadring and data exchange under different configurations to study the behaviour of the PoC components. Finally, we performed a theory-based assessment of the change in system quality from the existing X-Road implementation to the PoC implementation. In this section, we answer the research question **[RQ4] What is the viability of the proposed proof-of-concept implementation for X-Road with decentralized public key infrastructure?** It is broken down into two sub-research questions:

- RQ4.1 Does the proof-of-concept implementation meet the functional requirements?
- RQ4.2 How does the X-Road system quality change with DPKI in comparison to PKIX?

5.1 Setup

An X-Road network instance has been set up with the PoC implementation for evaluating the functional requirements. To become a member of this instance, organization must hold a business registration certificate as verifiable credential in its SSI agent. In the network there should be a single Central Server and two Security Servers. There should be a separate SSI agent (ACA-Py instance) for every Central Server and Security Server.

Two components have been implemented to assist with the evaluation. The first component is a web service that has been implemented with Java as the service provider's information system. It exposes REST APIs for adding and retrieving book records. The second component is a web service that automatically issues credentials as requested. It consists of a controller server implemented with Java and an ACA-Py instance. Two types of credentials can be requested. The first credential type is named ”Business Registration Certificate” and will be used for evaluating onboarding-related functional requirements. It contains three attributes, namely ”name”, ”registryCode”, and ”established”. The second credential type is named ”Startup Estonia Membership” and will be used for evaluating access control-related functional requirements. It contains three attributes, namely ”name”, ”membershipCode”, and ”startDate”.

5.2 Evaluation of Functional Requirements

In this section, we evaluate the PoC implementation of X-Road components against the functional requirements outlined in Section 3.4. We use the described acceptance criteria

to determine if each requirement has been met.

5.2.1 Evaluation of Common Functional Requirements of X-Road Components

Common functional requirements of X-Road components described in Table 7 are related to the use of VP. In the PoC implementation, X-Road components control ACA-Py instances to generate, request and verify AnonCreds presentations. Thus these common functional requirements are accomplished through the use of ACA-Py and AnonCreds.

Functional requirement 1

Acceptance criteria: A VP created by a component can be cryptographically verified that it is created by the holder of VC.

Evaluation details: AnonCreds presentation contains proof of credential-holder binding through the use of link secrets. Verifiers that can process AnonCreds presentation can verify the VP is created by the holder of VC.

Validation result: Pass

Functional requirement 2

Acceptance criteria: A component cryptographically verifies that every received VP is issued to the prover.

Evaluation details: AnonCreds presentation contains proof of credential-holder binding through the use of link secrets. ACA-Py verifies the correctness of every proof received.

Validation result: Pass

Functional requirement 3

Acceptance criteria: A component cryptographically verifies that every received VC matches the properties in the presentation request using information obtained from the distributed ledger.

Evaluation details: AnonCreds presentation is generated to match the provided presentation request. ACA-Py verifies the correctness of every proof submitted by the prover by cross referencing the credential definition it collected from the verifiable data registry in the distributed ledger.

Validation result: Pass

Functional requirement 4

Acceptance criteria: A component cryptographically verifies the revocation status of every received VC. The revocation status can be obtained from the distributed ledger.

Evaluation details: AnonCreds presentation includes non-revocation proofs when non-revoked attributes are requested. ACA-Py verifies the correctness of the proof submitted by the prover by cross referencing the revocation status it collected from the revocation definition registry in the distributed ledger.

Validation result: Pass

Functional requirement 5

Acceptance criteria: A component can cryptographically verify a VP that is uniquely bound to the presentation request.

Evaluation details: To prevent replay attacks, each presentation request sent by ACA-Py includes a unique randomly generated number as nonce. ACA-Py verifies the same nonce exists in the received presentation.

Validation result: Pass

5.2.2 Evaluation of Functional Requirements of Central Server

In this section, we evaluate the implementation of the Central Server against its functional requirements.

Functional requirement 6

Acceptance criteria: Central Server administrators can change the attributes in the VP request for onboarding purposes.

Evaluation details: Central Server administrators can customize the VP request by setting the environment variable "XROAD_PROOF_REQUEST" to a valid AnonCreds presentation request in JSON format when starting the Central Server. Listing 6 shows the AnonCreds presentation request in JSON format used in the Central Server configuration

```
{
  "name": "Proof of Identity",
  "version": "1.0",
  "requested_attributes": {
    "0_business_uuid": {
      "names": ["name", "registryCode", "established"],
      "restrictions": [{
        "schema_name": "Business Registration Certificate"
      }]
    }
  },
  "requested_predicates": {}
}
```

Listing 6. Onboarding Proof Request

for evaluation. It is sent to the Security Server after the Security Server sends a DID registration transaction endorsement request to the Central Server.
Validation result: Pass

Functional requirement 7

Acceptance criteria: Central Server endorses DID registration transactions submitted by a Security Server which has submitted a valid VP. Central Server does not endorse DID registration transactions submitted by a Security Server which has not submitted a valid VP.

Evaluation details: After receiving a DID registration transaction endorsement request from a Security Server, Central Server checks if the Security Server has submitted a valid VP for onboarding. Only if the Security Server has submitted a valid VP for onboarding Central Server will endorse the DID registration transaction and return it to the Security Server. Otherwise, Central Server rejects the DID registration transaction endorsement request.

Two Security Servers have been set up to evaluate the requirement. They are simulated to be under the control of two distinct organizations interested in joining the X-Road instance. One of these Security Servers is connected to an ACA-Py instance that holds the required VC in the test scenario, namely ”Business Registration Certificate”. The DID registration transaction endorsement request sent by this Security Server is successfully accepted and processed by the Central Server. The newly registered DID

can be verified on the ledger browser. In contrast, the other Security Server which is connected to an ACA-Py instance without the required credential, has its DID registration transaction endorsement request rejected by the Central Server.

Validation result: Pass

5.2.3 Evaluation of Functional Requirements of Security Server

In this section, we evaluate the implementation of the Security Server against its functional requirements.

Functional requirement 9

Acceptance criteria: Security server administrators can set the attributes in the VP request for access control before starting the server.

Evaluation details: Security Server administrators can customize the VP request by setting the environment variable "XROAD_PROOF_REQUEST" to a valid AnonCreds presentation request in JSON format when starting the Security Server. When the environment variable is not set, the Security Server sends the default proof request for access control, same as listing 6. After the environment variable is set to a modified AnonCreds presentation request and the Security Server is restarted, the Security Server sends the updated proof request for access control. Listing 7 shows the modified AnonCreds presentation request in JSON format.

Validation result: Pass

Functional requirement 10

Acceptance criteria: Security Server clients can send DID registration transaction endorsement requests to the Central Server using an API provided by the Security Server.

Evaluation details: Security Server client can make a HTTP POST request to Security Server's endpoint '/management/onboard' to send DID registration transaction endorsement requests to the Central Server.

Validation result: Pass

```
{
  "name": "Proof of Identity and Startup Membership",
  "version": "1.0",
  "requested_attributes": {
    "0_business_uuid": {
      "names": ["name", "registryCode", "established"],
      "restrictions": [{
        "schema_name": "Business Registration Certificate"
      }]
    },
    "0_startup_uuid": {
      "names": ["name", "membershipCode"],
      "restrictions": [{ "schema_name": "Startup Estonia Membership" }]
    }
  },
  "requested_predicates": {
    "0_startDate_GE_uuid": {
      "name": "startDate",
      "p_type": ">=",
      "p_value": 20200101,
      "restrictions": [{ "schema_name": "Startup Estonia Membership" }]
    }
  }
}
```

Listing 7. Custom Proof Request for Access Control

Functional requirement 11

Acceptance criteria: Security Server clients can send messages to service providers in the same X-Road network using an API provided by the Security Server.

Evaluation details: Security Server client can send HTTP requests to service providers using its Security Server's endpoint "/x-road/{information system's DID}/*".

Validation result: Pass

Functional requirement 12

Acceptance criteria: Security Server sends a VP request for access control to the other member's Security Server after receiving a message from its client targeted to another member and it has no record of valid VPs sent by the other member's Security Server.

Evaluation details: In the implementation, after receiving a message from its client targeted to another member, Security Server checks if there is a valid VP sent by the other member's Security Server. If there is no valid VP found, Security Server sends the configured AnonCreds presentation request for access control, to the other member's Security Server via their respective ACA-Py instances.

To test the requirement, a message was sent to a service consumer's Security Server targeted to a service provider for the first time. The service consumer's Security Server sent the AnonCreds presentation request for access control to the service provider's Security Server.

Validation result: Pass

Functional requirement 13

Acceptance criteria: Security Server transmits messages from its client to another Security Server which has submitted a valid VP. Security Server does not transmit messages from its client to another Security Server which has not submitted a valid VP.

Evaluation details: In the implementation, Security Server only invokes the non-repudiable message exchange protocol implemented in ACA-Py to transmit messages from its client to another Security Server if it verifies that a cached valid VP submitted by the other Security Server is found or a fresh VP is responded from the other Security Server.

Two tests were performed for the evaluation. First, a message was sent to a service

consumer's Security Server targeted to a service provider whose Security Server could construct a valid VP that fits the VP request. The message was received by the service provider's Security Server. Then, a message was sent to a service consumer's Security Server targeted to another service provider whose Security Server could not construct a valid VP that fits the VP request. The service provider did not receive the message.

Validation result: Pass

Functional requirement 14

Acceptance criteria: The signature sent along with a message can be verified that it was computed from the message and a private key associated with the sender's DID.

Evaluation details: Security Server uses the non-repudiable message exchange protocol implemented in ACA-Py for data exchange. Security Server instructs ACA-Py to sign the message with a verification key associated with the sender's public DID registered on the distributed ledger.

Validation result: Pass

Functional requirement 15

Acceptance criteria: Security Server transmits messages from another Security Server to its client if the message signature is valid. Security Server does not transmit messages from another Security Server to its client if the message signature is invalid.

Evaluation details: Security Server uses the non-repudiable message exchange protocol implemented in ACA-Py for data exchange. ACA-Py verifies the message signature as part of the protocol implementation. Only if the signature is valid ACA-Py will forward the message to its controller, in this case the Security Server. Security Server then transmits the messages to its client.

Validation result: Pass

5.3 Change in System Quality

Using the assessment model proposed by [42], we evaluate the changes in system quality that define the trustworthiness of the X-Road. The quality properties and measures from the assessment model are presented in Figure 11. Table 10 shows the overview of the evaluation results.

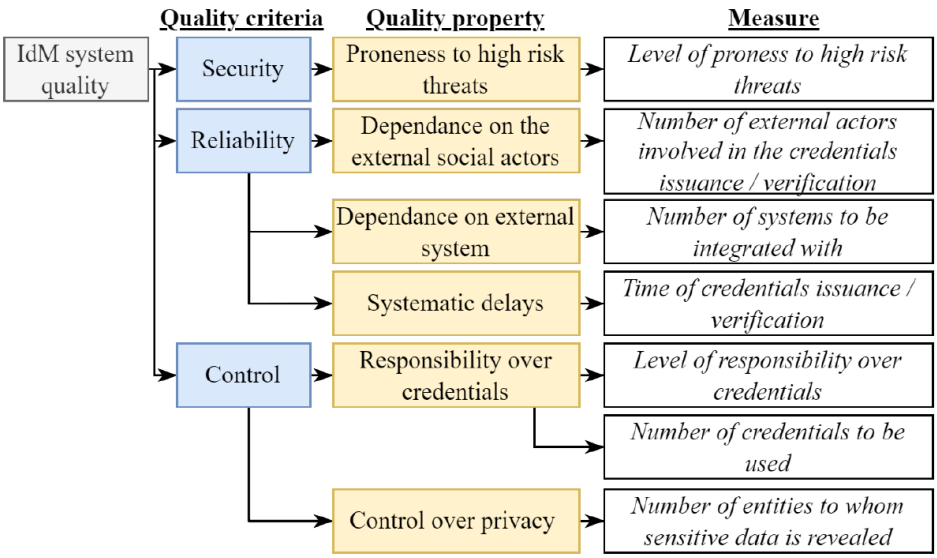

Figure 11. Identity management system quality assessment model [42]

Table 10. Change in System Quality

Quality Criteria	***Quality Property***	***Measure***	***Change***
Security	Proneness to high risk threats	Level of proneness to high risk threat	Decrease
Reliability	Dependance on the external social actors	Number of actors involved in the credentials issuance/verification	Unchanged
	Dependance on external system	Number of systems to be integrated with for issuance/verification	Unchanged
	Systematic delays	Time of credentials issuance/verification	Decrease by the time required for credential issuance due to key rotation
Control	Responsibility over credentials	Level of responsibility over credentials	Increase
		Number of credentials to be used	Unchanged
	Control over privacy	Number of entities to who sensitive data is revealed during issuance/verification	Increase

First, we assess security based on the proneness to high-risk threats. The PoC system performs credential revocation checks using a revocation status list distributed across decentralized nodes of the ledger. This approach reduces the computational demand on any single machine, making it more difficult for denial-of-service attacks to occur. Additionally, the system generates fresh non-revocation proof instead of relying on cached proof, such as the cached OCSP responses used in existing X-Road. This approach enables the use of mechanisms like nonces to prevent replay attacks. Thus, the PoC has a higher security level than the existing X-Road.

Second, we evaluate the reliability of the PoC based on the dependence on external social actors, dependence on external systems, and systematic delay. The reliability of the system increases as the values of these measures decrease. While existing X-Road depends on CAs to issue and verify certificates, the PoC depends on credential issuers to issue and update certificate revocation status. In turn, the dependence on external social actors remains unchanged. While the PoC introduces a dependency on SSI agents, it removes the dependency on CAs, and hence, the dependence on external systems remains unchanged. Finally, the systematic delay in the PoC decreases because members can self-rotate their signing keys proactively without needing to request a new credential, reducing the overall time of credentials issuance.

Third, we assess control over the credentials the identity holder has based on the level of system responsibility over credentials and the level of privacy control the Member has over its identity. The level of responsibility over credentials in the PoC decreases as the credentials and key materials are managed by external SSI agents. On the other hand, the level of privacy control that a member has over its identity increases due to the use of credential implementation that supports selective disclosure and predicates.

5.4 Answers to Research Questions

In this section, we evaluated the PoC implementation against the function requirements elicited in Section 3 and assessed the change in system quality from X-Road with PKIX to X-Road with DPKI to answer the research question *[RQ4] What is the viability of the proposed proof-of-concept implementation for X-Road with decentralized public key infrastructure?* It was broken down into two sub-research questions and we gave answers to each of them.

[RQ4.1] Does the proof-of-concept implementation meet the functional requirements? The PoC implementation satisfies all the acceptance criteria of the functional requirements described in Section 3. Thus, the PoC implementation meets all the functional requirements.

[RQ4.2] How does the X-Road system quality change with DPKI in comparison to PKIX? The PoC system exhibits enhanced security and reliability compared to the existing X-Road. Members in the PoC have greater control over their credentials.

6 Concluding Remarks

This thesis aims to answer the main research question: **[MRQ] How can trust be established between information systems using a decentralized public key infrastructure?** By following the Design Science Research Method (DSRM), we first studied traditional public key infrastructure with X.509 certificates (PKIX) and its limitations, as well as emerging Self-Sovereign Identity (SSI) technologies like decentralized identifiers (DIDs) and verifiable credentials (VC). We also examined the usage of PKIX in member onboarding and access control in the data exchange system X-Road. Subsequently, we conducted a literature review to explore related work that uses SSI technologies to facilitate onboarding and access control.

Our findings revealed that existing studies on SSI focus on enterprise-to-end-user use cases, and there has been limited research on the enterprise-to-enterprise applications of SSI technologies. To address this gap, we suggested an alternate architecture for X-Road to migrate to a DPKI constructed with DIDs and VCs to explore potential benefits and limitations.

To design DPKI with DIDs and VCs for X-Road, we established a set of design goals and use cases that require trust establishment. We then elicited a list of functional requirements for X-Road components. Afterward, we implemented a proof-of-concept (PoC) using open-source SSI technologies from the Hyperledger Indy ecosystem, including the Hyperledger Indy ledger, the Hyperledger Aries Agent, the Sorvin DID method, and the AnonCred verifiable credentials.

We built a simplified X-Road Security Server and a simplified X-Road Central Server and demonstrated the trust establishment process for member onboarding and access control using SSI technologies instead of PKIX. The PoC was evaluated based on the elicited functional requirements. The validation results showed that the PoC fulfilled all the functional requirements. The changes in system quality that define trustworthiness were also assessed using the assessment model proposed in preliminary work [42]. The results showed that the PoC improved security and reliability of the system, and members of the PoC gained greater control over their credentials.

6.1 Answer To Research Question

The answer to the main research question is that decentralized identifiers and verifiable credentials can be combined to construct a decentralized public key infrastructure and replace X.509 certificates in PKIX. Decentralized identifiers can be used to bind an identifier to public keys while verifiable credentials can be used for asserting information about the subject of the identifier. The use of DID improves the subject's control over its public key as it enables key rotation without intermediaries. The use of VC enables features like selective disclosure and predicates, which make the trust establishment process more flexible. However, scalability issues for credential revocation status checks

still exist, as they do in PKIX. Using DLT-based VCs helps alleviate those problems by reducing the load of a single machine.

6.2 Limitations

Our study have some limitations. First, our design may be biased towards the SSI implementations in the Hyperledger Indy ecosystem. Since SSI is an emerging field, there are no established architectures for enterprise software to integrate with the SSI model. While our study and most studies from the literature review use technologies from the Hyperledger Indy ecosystem as a reference [28, 29, 30, 32], the high-level standard of decentralized identifiers and verifiable credentials may not enforce features supported by technologies in the Hyperledger Indy ecosystem. For example, not every verifiable credential implementation supports revocation and revocation status check with the ledger like AnonCreds does. Second, our design assumes that every X-Road member controls their own Hyperledger Aries instance as their SSI agent. In practice, it can be challenging to have all X-Road members have their own SSI agent that can communicate with each other.

6.3 Conclusion

Public key infrastructure with X.509 certificates has been the de facto standard for secure communication over the internet. The widespread adoption and long history of X.509 certificates have led to the development of a mature ecosystem of tools, libraries, and protocols that support its usage. This has made it a reliable and trusted mechanism for establishing secure connections between parties. However, PKIX is often accused of having a centralized trust model. Its validity checking mechanisms, Certificate Revocation Lists (CRL) and Online Certificate Status Protocol (OCSP), suffer from scalability issues due to its centralized nature, which leads to potential attacks such as denial-of-service attacks and replay attacks. These problems give rise to alternative decentralized public key infrastructures and Decentralized Identifiers (DIDs).

DIDs provide a mechanism for individuals to create and control their own unique identifiers without intermediaries, which can be used to authenticate themselves across different applications and systems. By definition, everyone can create a DID and write arbitrary information to the DID document. This characteristic is beneficial to individual privacy, as one can create as many identifiers as possible to represent different personas. However, it makes DIDs unsuitable for asserting information in a trustworthy way. In other words, a DID alone cannot be bound to a physical identity like a legal entity within a jurisdiction, such as an X-Road member. To do so, verifiable credentials are needed.

Although DIDs are designed to enable decentralized systems without reliance on centralized authorities, the use of verifiable credentials reintroduces a degree of centralization through the need for trusted issuers of credentials. In fact, verifiable credentials,

especially the revocable one, share similarities with certificates. First, credential issuers, like certificate authorities, are trusted actors in a system. In turn, systems depending of verifiable credentials and their issuers have the same single-point-of-failure as traditional PKI does. If a trusted issuer is compromised, the system that depends on it halts. Not only does the credential issuance process stop, existing revocable credentials can no longer be used since there is no way to obtain the latest revocation status. Second, issuers of revocable credentials have to provide credential status checking mechanisms to verifiers. However, the protocols for the mechanisms are not defined in the verifiable credential data model [8]. That means a centralized server like an OCSP responder can be deployed, or a credential revocation list like the a certificate revocation list can be used. These mechanisms are expected to suffer similar scalability issues CRL or OCSP have. In the PoC of the study, the credential revocation list is written on a distributed ledger. While it reduces the load of a single machine, it cannot provide real time revocation status check. On the other hand, traditional PKI with X.509 can distribute the revocation status using distributed ledger technology to reduce the load of a single CRL Distribution Point. In other words, the problem of revocation status checking mechanisms is not specific to certificates, but is due to the centralized nature of these mechanisms, which also exists in the use of verifiable credentials.

Despite the similarities, there are lessons to be learned from the development of DIDs and VCs that could improve traditional PKI. First, DIDs allow their owners to bind public keys for verification to a unique identifier. Consequently, the owner can proactively rotate the keys without changing the identifier as a cybersecurity best practice. If an X.509 certificate binds a physical identity to a DID instead of a public key, the certificate subject could rotate the public key without interacting with the CA. Second, verifiable credentials are designed in such a way that they can be presented with or without the original credential, while the authorship of the data can still be trusted after a process of cryptographic verification. This is different from the use of X.509 certificates, which are shared as is with the verifier when presenting. The concept of verifiable presentation enables features like selective disclosure and predicates, which make the trust establishment process more flexible. Since X.509 certificate is simply a container that holds a list of key-value pairs, in theory it can be extended to be presented as a verifiable presentation of a verifiable credential to improve its flexibility. Third, PKIX can learn from DLT-based revocation mechanisms like the one in AnonCreds to provide a scalable and privacy-preserving way for certificate validity checks. When choosing a DLT, it is important to consider factors such as the number and the decentralization of validator nodes, consensus mechanisms, time to finality etc. By taking these lessons into account, traditional PKI can enhance its resiliency and trustworthiness while retaining its existing benefits.

6.4 Future work

Future work could explore the integration of decentralized identifiers (DIDs) into X.509 certificates as additional attributes to replace the subject-public-key binding. This integration would enable key rotation without requiring new certificates to be issued. Additionally, it may be beneficial to extend X.509 certificates to enable verifiable presentation without the original certificate, following the verifiable credential/presentation pattern. This extension would allow for more flexible use cases, such as sharing specific attributes of a certificate without revealing the entire certificate. These goals could be archived be defining new certificate extensions for X.509 certificates that contain DID information, as well as allowing X.509 certificates to be used as verifiable credentials.

References

[1] K. Cameron, "The laws of identity," *Microsoft Corp*, vol. 12, pp. 8–11, 2005.

[2] C. Huitema, D. Bachenheimer, D. O'Donnell, D. Reed, J. Bikoundou, J. Fleenor, K. Young, Kaliya Hand, J. Kneiss, Karl Jordan, L. Bendixsen, P. Subrahmanyam, S. Mukhopadhyay, S. Perry, V. Syntez, V. Malhotra, and W. Chu, "Introduction to Trust Over IP. Version 2.0.." `https://trustoverip.org/wp-content/uploads/Introduction-to-ToIP-V2.0-2021-11-17.pdf`. Last accessed 10-Apr-2023.

[3] R. Housley, W. Ford, W. Polk, and D. Solo, "Internet x. 509 public key infrastructure certificate and crl profile," tech. rep., Internet Engineering Task Force, 1999.

[4] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet x. 509 public key infrastructure certificate and certificate revocation list (crl) profile," tech. rep., Internet Engineering Task Force, 2008.

[5] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams, "X.509 internet public key infrastructure online certificate status protocol - OCSP," Request for Comments RFC 6960, Internet Engineering Task Force, 2013. Num Pages: 41.

[6] A. Preukschat and D. Reed, *Self-Sovereign Identity*. Manning Publications, 2021.

[7] D. Reed, M. Sporny, D. Longley, C. Allen, R. Grant, M. Sabadello, and J. Holt, "Decentralized identifiers (dids) v1.0," *W3C Working Draft*, 2020.

[8] Verifiable Credentials Working Group, "Verifiable Credentials Data Model v1.1." `https://www.w3.org/TR/vc-data-model/`. Last accessed 10-Apr-2023.

[9] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS Quarterly*, vol. 28, no. 1, pp. 75–105, 2004.

[10] C. Allen, A. Brock, V. Buterin, J. Callas, D. Dorje, C. Lundkvist, P. Kravchenko, J. Nelson, D. Reed, M. Sabadello, G. Slepak, N. Thorp, and H. T. Wood, "Decentralized public key infrastructure a white paper from rebooting the web of trust."

[11] M. Lodder and D. Hardman, "Sovrin DID Method Specification." `https://sovrin-foundation.github.io/sovrin/spec/did-method-spec-template.html`. Last accessed 10-Apr-2023.

[12] S. Curran, A. Philipp, H. Yildiz, S. Curren, and V. M. Jurado, "AnonCreds Specification." `https://hyperledger.github.io/anoncreds-spec/`. Last accessed 10-Apr-2023.

[13] The Sovrin Governance Framework Working Group, "Sovrin Glossary V3." https://sovrin.org/wp-content/uploads/Sovrin-Glossary-V3.pdf. Last accessed 10-Apr-2023.

[14] The Hyperledger Foundation, "Hyperledger Indy SDK." https://github.com/hyperledger/indy-sdk. Last accessed 10-Apr-2023.

[15] The Hyperledger Foundation, "Hyperledger Indy Plenum." https://github.com/hyperledger/indy-plenum. Last accessed 10-Apr-2023.

[16] S. Curran, P. Bastian, D. Hardman, C. Howland, C. Bormann, D. Wörner, D. Bluhm, and K. D. Hartog, "Indy DID Method." https://hyperledger.github.io/indy-did-method. Last accessed 10-Apr-2023.

[17] The Hyperledger Foundation, "Hyperledger indy roles and permissions." https://github.com/hyperledger/indy-node/blob/main/indy_common/authorize/auth_map.py. Last accessed 10-Apr-2023.

[18] The Hyperledger Foundation, "Hyperledger indy transaction endorser design." https://github.com/hyperledger/indy-node/blob/main/design/transaction_endorser.md. Last accessed 10-Apr-2023.

[19] W. Abramson, N. Hickman, and N. Spencer, "Evaluating trust assurance in indy-based identity networks using public ledger data," *Frontiers in Blockchain*, vol. 4, p. 622090, 2021.

[20] D. Hardman, "Aries RFC 0005: DID Communication." https://github.com/hyperledger/aries-rfcs/tree/070b325ef4a55005d79d03eb629f0399f3a0dfc4/concepts/0005-didcomm/README.md. Last accessed 10-Apr-2023.

[21] D. Hardman, "Aries RFC 0003: Protocols." https://github.com/hyperledger/aries-rfcs/blob/070b325ef4a55005d79d03eb629f0399f3a0dfc4/concepts/0003-protocols/README.md. Last accessed 10-Apr-2023.

[22] The Hyperledger Foundation, "Hyperledger Aries." https://github.com/hyperledger/aries. Last accessed 10-Apr-2023.

[23] D. Hardman, "Aries RFC 0004: Agents." https://github.com/hyperledger/aries-rfcs/blob/070b325ef4a55005d79d03eb629f0399f3a0dfc4/concepts/0004-agents/README.md. Last accessed 10-Apr-2023.

[24] Nordic Institute for Interoperability Solutions (NIIS), "X-Road Architecture." https://docs.x-road.global/Architecture/arc-g_x-road_arhitecture.html. Last accessed 10-Apr-2023.

[25] Nordic Institute for Interoperability Solutions (NIIS), "X-Road® History — X-Road® Data Exchange Layer." https://x-road.global/xroad-history. Last accessed 10-Apr-2023.

[26] Nordic Institute for Interoperability Solutions (NIIS), "X-Road Security Architecture." https://docs.x-road.global/Architecture/arc-sec_x_road_security_architecture.html. Last accessed 10-Apr-2023.

[27] M. Bakhtina, R. Matulevičius, A. Awad, and P. Kivimäki, *Rebooting Trust Management in X-Road*. Nordic Institute for Interoperability Solutions (NIIS), 2022.

[28] R. Soltani, U. T. Nguyen, and A. An, "Decentralized and privacy-preserving key management model," in *2020 International Symposium on Networks, Computers and Communications (ISNCC)*, pp. 1–7, IEEE, 2020.

[29] V. Schlatt, J. Sedlmeir, S. Feulner, and N. Urbach, "Designing a framework for digital kyc processes built on blockchain-based self-sovereign identity," *Information & Management*, vol. 59, no. 7, p. 103553, 2022.

[30] R. Belchior, B. Putz, G. Pernul, M. Correia, A. Vasconcelos, and S. Guerreiro, "Ssibac: self-sovereign identity based access control," in *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pp. 1935–1943, IEEE, 2020.

[31] N. Fotiou, E. Faltaka, V. Kalos, A. Kefala, I. Pittaras, V. A. Siris, and G. C. Polyzos, "Continuous authorization over http using verifiable credentials and oauth 2.0," *Open Identity Summit 2022*, 2022.

[32] Z. András Lux, D. Thatmann, S. Zickau, and F. Beierle, "Distributed-ledger-based authentication with decentralized identifiers and verifiable credentials," *arXiv e-prints*, pp. arXiv–2006, 2020.

[33] X. Fan, Q. Chai, L. Xu, and D. Guo, "Diam-iot: A decentralized identity and access management framework for internet of things," in *Proceedings of the 2nd ACM International Symposium on Blockchain and Secure Critical Infrastructure*, pp. 186–191, 2020.

[34] Nordic Institute for Interoperability Solutions (NIIS), "X-Road Security Server User Guide." https://www.x-tee.ee/docs/live/xroad/ug-ss_x-road_7_security_server_user_guide.html. Last accessed 10-Apr-2023.

[35] The Hyperledger Foundation, "Hyperledger Aries Cloud Agent Python." https://github.com/hyperledger/aries-cloudagent-python. Last accessed 10-Apr-2023.

[36] Province of British Columbia, "BC Digital Trust." https://digital.gov.bc.ca/digital-trust/. Last accessed 10-Apr-2023.

[37] Government of British Columbia Canada, "VON Network." https://github.com/bcgov/von-network. Last accessed 10-Apr-2023.

[38] R. West, D. Bluhm, M. Hailstone, S. Curran, S. Curren, and G. Aristy, "Aries RFC 0023: DID Exchange Protocol 1.0." https://github.com/hyperledger/aries-rfcs/blob/070b325ef4a55005d79d03eb629f0399f3a0dfc4/features/0023-did-exchange/README.md. Last accessed 10-Apr-2023.

[39] N. Khateev, S. Klump, and S. Curran, "Aries RFC 0453: Issue Credential Protocol 2.0." https://github.com/hyperledger/aries-rfcs/blob/070b325ef4a55005d79d03eb629f0399f3a0dfc4/features/0453-issue-credential-v2/README.md. Last accessed 10-Apr-2023.

[40] N. Khateev and S. Curran, "Aries RFC 0454: Present Proof Protocol 2.0." https://github.com/hyperledger/aries-rfcs/blob/070b325ef4a55005d79d03eb629f0399f3a0dfc4/features/0454-present-proof-v2/README.md. Last accessed 10-Apr-2023.

[41] K. D. Hartog, "Aries RFC 0066: Non-Repudiable Signature for Cryptographic Envelope." https://github.com/hyperledger/aries-rfcs/blob/070b325ef4a55005d79d03eb629f0399f3a0dfc4/features/0066-non-repudiable-cryptographic-envelope/README.md. Last accessed 10-Apr-2023.

[42] M. Bakhtina, R. Matulevičius, A. Awad, and P. Kivimäki, "On the shift to decentralised identity management in distributed data exchange systems," in *The 38th ACM/SIGAPP Symposium on Applied Computing (SAC '23), March 27-March 31, 2023, Tallinn, Estonia*, (New York, NY, USA), Association for Computing Machinery, 2023.

Acronyms

ACA-Py Hyperledger Aries Cloud Agent - Python. 34–37, 39, 41, 43–47, 49, 50

CA Certificate Authority. 2, 6, 8–10, 12, 15, 18, 52, 56

CRL Certificate Revocation List. 6, 9, 56

CS X-Road Central Server. 20

DID Decentralized Identifier. 6, 10–19, 21, 23, 27, 28, 30–32, 34, 39, 42, 46, 47, 49, 50, 54–57

DIDcomm Decentralized Identifier Communication. 13, 34, 42

DLT Distributed Ledger Technology. 2, 6, 10, 12, 15, 55, 56

DPKI Decentralized Public Key Infrastructure. 2, 3, 6, 7, 9, 10, 17, 18, 29, 43, 52–54

DSRM Design Science Research Method. 7

HTTP Hypertext Transfer Protocol. 13, 16, 36, 37, 39, 41, 42, 47, 49

IDP Identity Provider. 10

IRC Internet Relay Chat. 13

IS Information System. 24, 25

MITM Man-in-the-middle attack. 10

OCSP Online Certificate Status Protocol. 6, 9, 15, 18, 52, 56

PKI Public Key Infrastructure. 16, 17, 56

PKIX Public Key Infrastructure with X.509. 2, 3, 6–10, 12, 14, 17, 18, 43, 52–56

PoC Proof of concept. 7, 30–35, 37, 39–41, 43, 44, 52–54, 56

SS X-Road Security Server. 22

SSI Self-Sovereign Identity. 7, 10, 12, 13, 15–20, 22, 29–32, 34, 42, 43, 52, 54, 55

SSL Secure Sockets Layer. 8

TLS Transport Layer Security. 8, 13

UI User Interface. 23, 32, 37

URI Uniform Resource Identifier. 10

VC Verifiable Credential. 6, 12, 14–17, 19, 26, 31, 44–46, 54–56

VDR Verifiable Data Registry. 14

VP Verifiable Presentation. 12, 16, 21, 23, 24, 26–28, 44–47, 49, 50

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Kin Long Leung**,
(author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   A Decentralized Public Key infrastructure for Trust Management in X-Road,
   (title of thesis)

   supervised by Mariia Bakhtina, Ahmed Awad and Raimundas Matulevičius.
   (supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Kin Long Leung ***09/05/2023***