CYBERNETICA

Interoperability Platform Architecture for Large Digital Governments

Technical Report

1. Introduction

The implementation of digital government interoperability platforms in service-intensive large countries presents unique challenges and opportunities. This report delves into the architectural and operational considerations necessary to scale these platforms effectively while maintaining performance and security. By addressing these complexities, the research aims to provide actionable insights for adapting interoperability solutions to meet the demands of large-scale e-government environments.

Cybernetica exports digital government solutions to many countries in the world. The solutions include SplitKey technology for eID, UXP technology for interoperability and low-code tools for government digital services. Some countries where Cybernetica aims to export digital government interoperability platform UXP, are considerably larger compared to Estonia or Finland in number of citizens, area etc. This often translates from one hand to more complex administrative apparatus and from the other hand to higher volume data exchange transactions. When large countries reach to the same service intensity of digital government for example Estonia, it poses new challenges for the digital government interoperability platform.

2. References

- [ACME] Internet Engineering Task Force. Automatic Certificate Management Environment (ACME). RFC 8555, 2019.
- [ARC-PATTERN] The Open Group. Architecture Patterns. https://pubs.opengroup.org/architecture/togaf92-doc/arch/chap22.html, 2018 (accessed 04.12.24).
- [BATCH-HASH] Margus Freudenthal. Using Batch Hashing for Signing and Time-Stamping. Research Report T-4-20, Cybernetica AS, 2013.
- [DESI] The European Commission. The Digital Economy and Society Index (DESI). https://digital-strategy.ec.europa.eu/en/policies/desi, 2022 (accessed 22.11.24).
- [DIG-LIT] Agnieszka Agata Tomaszewicz. The impact of digital literacy on e-government development. Online Journal of Applied Knowledge Management, Vol. 3, No. 2, pp. 45–53. https://www.iiakm.org/ojakm/articles/2015/volume3_2/OJAKM_Volume3_2pp45-53.pdf, 2015 (accessed 01.12.24).
- [DIG-SKILLS] The World Bank. Digital Skills: Frameworks and Programs, https://documents1.worldbank.org/curated/en/562351611824219616/pdf/Digital-Skills-Frameworks-and-Programs.pdf, 2020 (accessed 01.12.24).
- [DGIPRA] Keegan McBride, Sujani Kamalanathan, Sandhra-Mirella Valdma, Taavi Toomere, and Margus Freudenthal. Digital Government Interoperability Platform Reference Architecture. https://cyber.ee/uploads/Digital_Government_Interoperability_Platform_Reference_Architecture_4d2b8a83f6.pdf, 2022 (accessed 04.12.24).
- [DGSTRAT] OECD. Recommendation of the Council on Digital Government Strategies.

https://legalinstruments.oecd.org/en/instruments/OECD-LEGAL-0406, 2014 (accessed 04.12.24).

- e-Estonia. Why there's no digital transformation without interoperability. https://e-estonia.com/why-theres-no-digital-transformation-without-interoperability/, 2021 (accessed 04.12.2024).
- [EGDI] The United Nations. UN E-Government Survey 2024. https://publicadministration.un.org/egovkb/en-us/Reports/UN-E-Government-Survey-2024, 2024 (accessed 04.12.24).
- [EIF] EU. EIF Conceptual Model. https://ec.europa.eu/isa2/sites/default/files/eif_leaflet_final.pdf, 2017 (accessed 19.12.24).
- [GOV-RESIL] Andres Raieste, Andri Rebane, Madis Tapupere, Dr. Keegan McBride. Government Resilience in the digital age, chapter 01. Adopt a digital-first approach to critical public services and registries. https://info.nortal.com/hubfs/Gov_Resilience_web.pdf?hsLang=en, 2024 (accessed 13.12.2024).
- [GROW-POP] [Cătălina Mărcuță & MoldStud Research Team. Building Scalable E-Government Apps for Growing Populations. https://moldstud.com/articles/p-building-scalable-e-government-apps-for-growing-populations, 2024 (accessed 25.11.24).
- [HIGH-PERF] Arne Ansper, Ahto Buldas, Margus Freudenthal, and Jan Willemson. High Performance Qualified Digital Signatures for X-Road. In 18th Nordic Conference, NordSec 2013, LNCS 8208, pages 123–138. Springer, 2013.
- [HP-PROFILE] Margus Freudenthal. Profile for High-Performance Digital Signatures. Research Report T-4-23, Cybernetica AS, 2017.
- [INTEROP-ARC] Taavi Toomere. Interoperability Architecture for Digital Government Organization, https://cyber.ee/uploads/Interoperability_Architecture_for_Digital_Government_Organization_1_0_D_2_522_v1_1_e6b3c2919e.pdf, 2023 (accessed 13.12.24).
- [ISA2] ISA2 programme. The new European interoperability framework. https://ec.europa.eu/isa2/sites/default/files/eif_brochure_final.pdf, 2017 (accessed 06.11.24).
- [LARGE-COUNTRY] Worldometer. Population by country. https://www.worldometers.info/world-population/population-by-country/, 2024 (accessed 29.11.24).
- [TOGAF] The Open Group. The TOGAF Standard, Definitions. https://pubs.opengroup.org/architecture/togaf9-doc/arch/chap03.html#tag_03_23, 2018 (accessed 04.12.24).
- [WoG] Whole of Government Approach, https://en.wikipedia.org/wiki/Whole-of-government_approach, 2024 (accessed 19.12.24).

2.1. Unified eXchange Platform (UXP)

UXP is a Cybernetica´s interoperability platform technology that enables the implementation of government digital services using the Whole of Government approach [WoG]. UXP was forked from X-Road in 2016 and has since then developed independently by Cybernetica.

The main characteristics of UXP include:

Central functions of Governing Authority

- Governance of interoperating member organisations
  - PKI infrastructure is used for authentication of member organisations and e-seals of member organisations
  - UXP Registry is used for member management
  - UXP Directory is used for public discovery of organisations in the network
- Governance of data provider and data consumer IT systems
  - UXP Registry is used for management of IT systems
  - UXP Monitoring is used for near-realtime monitoring of client IT system activities
  - UXP Directory is used for public discovery of IT systems
- Governance of member UXP Security Servers
  - UXP Registry is used for management of the member UXP Security Servers
  - UXP Monitoring is used for near-realtime monitoring of UXP Security Server activities
- Management of network security policy
  - UXP Registry is used for the management of Certification Authorities (CA)
  - UXP Registry is used for the management of Timestamping Authorities (TSA)
- Distribution of UXP network configuration to members
  - UXP Registry is used for distribution of the UXP network configuration and security policy to the member UXP Security Servers

Decentralised functions of UXP member organisations

- Governance of APIs
  - UXP Security Server is used for the registration of the APIs in the UXP network. SOAP and REST web services are supported
  - UXP Monitoring is used for near realtime monitoring of API activities
  - UXP Directory is used for public discovery of APIs and the API usage statistics
  - UXP Security Server is used for API access control
  - UXP Security Server (Verifier component) is used for verification of the signed and timestamped messagelog
- Peer-to-peer secure data exchange between member organisations. UXP Security Server is used for:
  - Validating certificates of the client system
  - Applying the security policy: encrypting and decrypting the message using mTLS, signing it with the organisation sign certificate and signing it with the UXP Security Server auth certificate.

- Validating certificates of communicating UXP Security Server and member organisation
- routing and synchronous peer-to-peer data exchange
- Creating signed and timestamped messagelog
- Validating service access rights

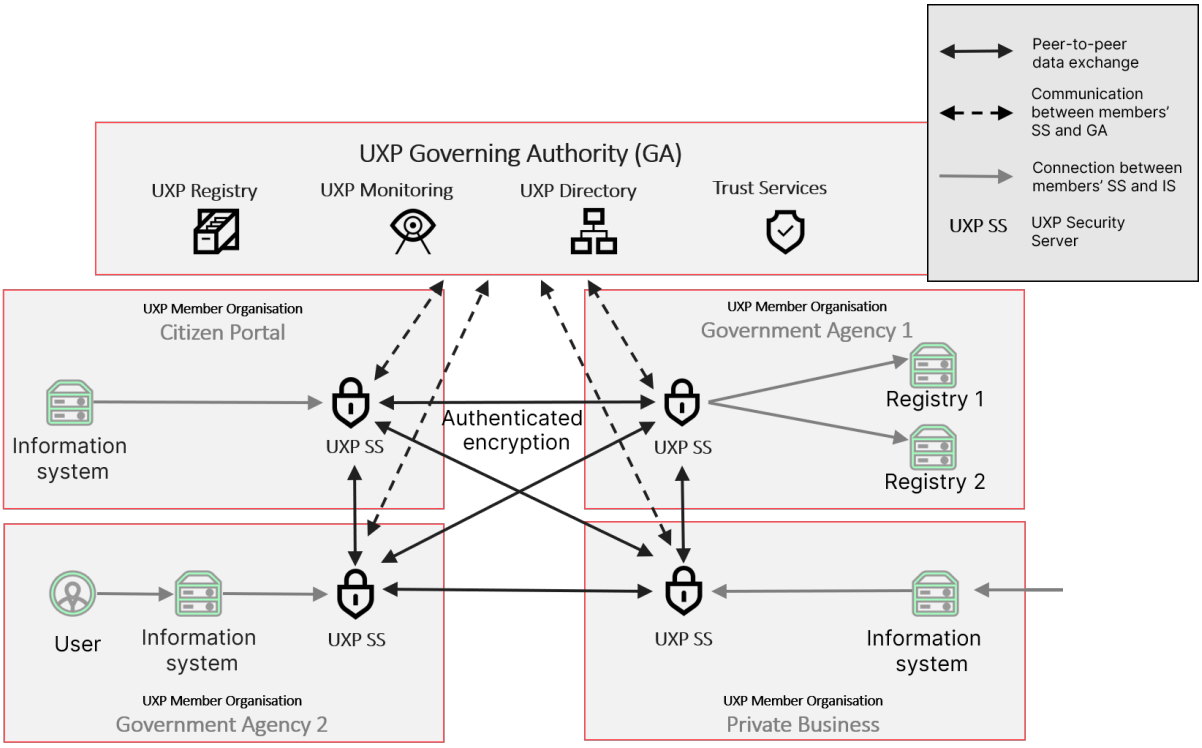

Figure 1. Overview of UXP

2.2. Research question

Primary research question:

- What architectural patterns and design principles enable optimal scaling of the UXP interoperability platform for service-intensive large countries while maintaining security and performance?

Secondary research question

- How population size and maturity of digitalization impact the Interoperability Platform

2.3. Research domain

The research domain is the performance and scalability of the digital government.

European Interoperability Framework Conceptual Model [EIF] offers a non-technical view of the provision of integrated public services.

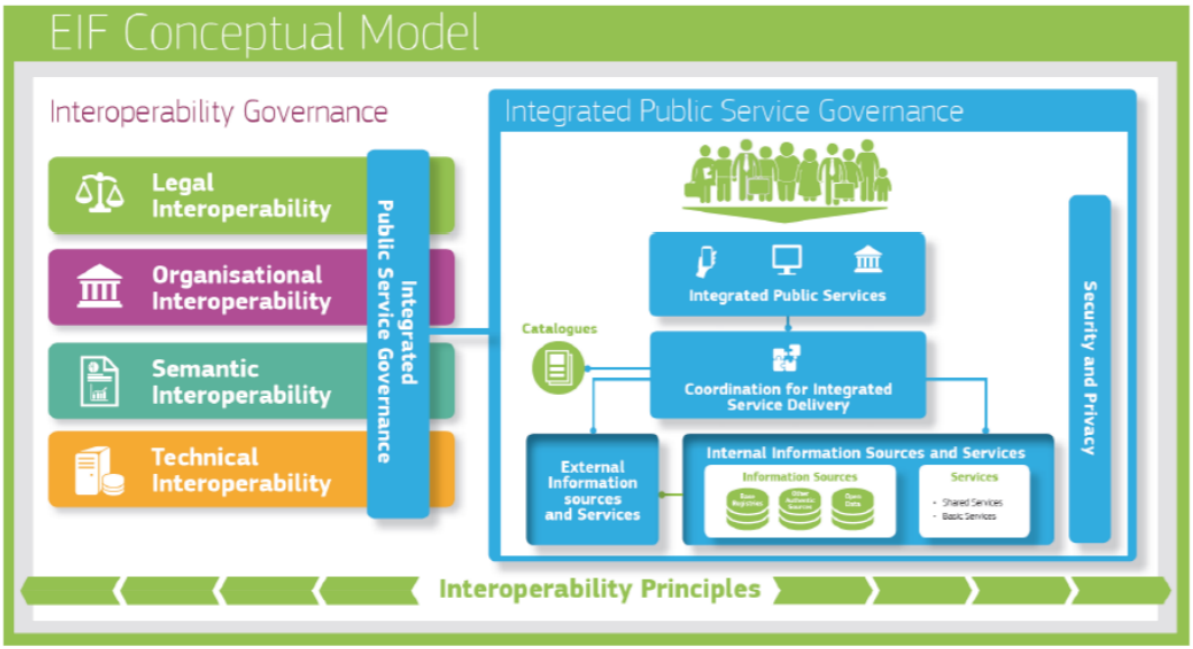

Figure 2. EIF interoperability conceptual view

For purpose of this report, we use a more technical architectural view based on building blocks. This is a specific minimal version of Cybernetica´s digital government reference architecture [DGIPRA]. In the simple model digital government architecture consists of the following building blocks:

1. Portals: such as citizen portal
2. Business logic: integrated public services
3. Platform tools: interoperability platform, digital identity
4. Distributed data: data such as base registries and different data sources

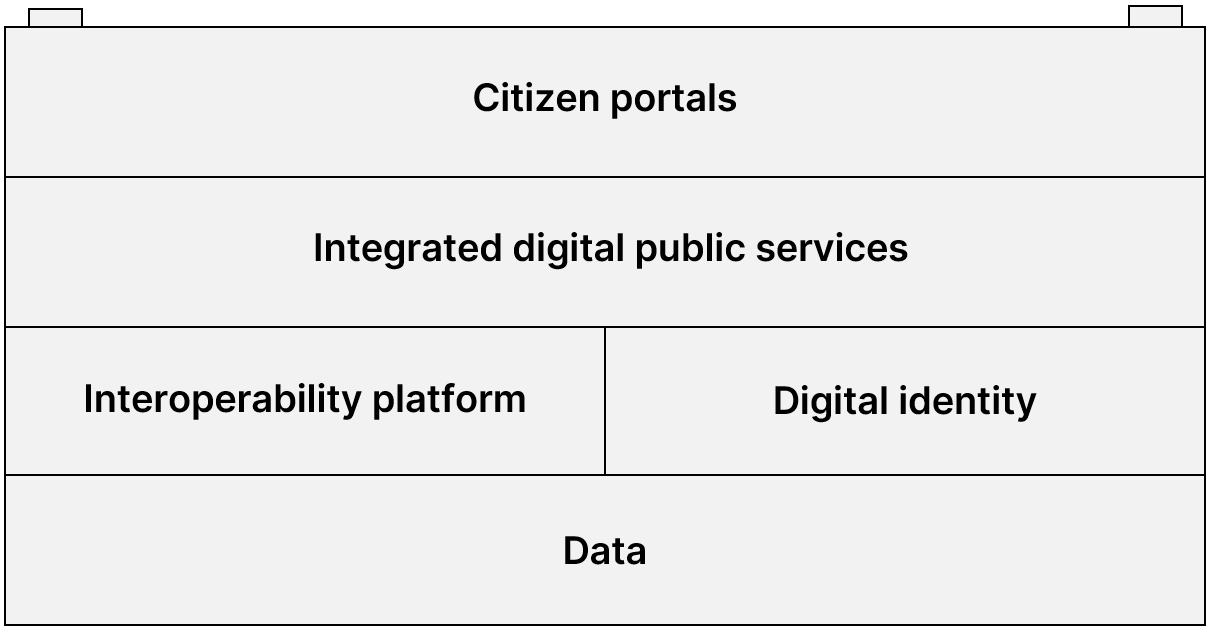

Figure 3. Simple digital government architecture

2.4. Impact

This research will enable Cybernetica to:

- Advise customers to design the digital government architecture while avoiding the scalability issues
- Open up the new market segment of large countries for UXP
- Redesign UXP to architecture to support the implementation in service-intensive large countries

2.5. Methodology

1. Desk research of external factors that drive performance and scalability requirements of the interoperability platform
2. Desk research of optimisation strategies for performance and scalability of government IT systems
3. Architectural design of the UXP based on the desk research of optimisation strategies

2.6. Key concepts

Interoperability is defined as either the ability to share information and services, or the ability of systems or components to exchange and use information or provide and receive services from other systems [TOGAF].

Digital government as defined by the OECD, “refers to the use of digital technologies, as an integrated part of governments’ modernisation strategies, to create public value. It relies on a digital government ecosystem comprised of government actors, non-governmental organisations, businesses, citizens’ associations and individuals which supports the production of and access to data, services and content through interactions with the government" <<DGSTRAT>.

A digital government interoperability platform is a building block for the digital transformation of public administrations. These interoperability platforms “allow public and private sector entities to control which external parties get access to their databases securely” [E-EST]. A digital government interoperability platform defined in a narrow sense may include just government institutions, but the largest positive network effects are created when the platform is also open to other organizations such as the private sector and third parties.

In the context of the Conceptual model for integrated public services in the European Interoperability Framework (EIF), the interoperability platform is depicted as a "Coordination for Integrated Service Delivery" building block [ISA2].

Architectural pattern is a way of putting building blocks into context; for example, to describe a reusable solution to a problem [ARC-PATTERN].

Large country. In this paper we consider large country to be one with a population of greater

than 20 million people. 65 countries in the world have a larger population than 20 million people which is the top 27% of most populous countries in the world [LARGE-COUNTRY].

Service-intensive digital government. In this paper we consider the service-intensive digital government to be a digital government that has intensive use of digital services and thus, high number of transactions in the interoperability platform. This can be for example measured by the E-government Development Index (EGDI) value [EGDI]. The leading countries of the index, Denmark, Estonia and Singapore, are good examples of such countries.

3. Aspects driving performance requirements of interoperability platform

In this section, we analyze the implications of external factors on the performance and scalability requirements of the interoperability platform. The analysis is conducted ceteris paribus, focusing on how each specific factor impacts usage while assuming that all other factors remain constant.

3.1. Population

The external factors that affect performance and scalability requirements of interoperability platform:

Population size creates direct pressure on the end-user capacity of the digital government including the interoperability platform [GROW-POP].

Digital literacy of citizens. The more citizens use digital tools the more they can potentially use digital government services. The digital skills are measured eg. by the human capital dimension of the Digital Economy and Society Index (DESI) [DESI], [DIG-LIT].

Uptake of digital government services. The more citizens prefer to use government services over digital channels the the more they also use digital government services and thus, the interoperability platform. The use of digital government services by different EU countries is measured by the "public service index" dimension of the Digital Economy and Society Index (DESI) [DESI].

All those factors imply requirements for higher performance and better scalability due to :

- Large number of transactions, both initiated by the citizens themselves and by government officials/backend processes
- Large number of records in different databases and registries
- Large number of transactions in the digital ID service
- Large number of businesses

3.2. Organisational complexity

This section is based on the research on digital government organisation topology, political order and sovereignty of organisations [INTEROP-ARC].

The volume of data exchange transactions depends on the number of organisations that operate in a policy domain. A policy domain can be managed by different numbers of organisations in different countries. The more autonomous organisations there are, the more data exchange transactions there will be between them.

The external factors that affect performance and scalability requirements of interoperability platform:

Political order affects government organisations' vertical and horizontal structure and autonomy. Several vertical levels of governance lead to more organisations and possible duplication of work on different levels. Horizontal fragmentation of policy domains leads to a higher number of organisations.

Complexity of government administrative structure affects the need for backoffice communication between the government organisations.

High autonomy of government authorities leads to stronger organisation borders and more formalised data exchange between the organisations.

Fragmentation of non-government organisations affects the number of private and 3rd sector organisations that communicate with the government.

All those factors imply requirements for higher performance and better scalability due to:

- More communication between government organisations in horizontal and vertical levels
- Big workload for managing, and registering organizations

3.3. Digitalisation

The external factors that affect performance and scalability requirements of interoperability platform:

Higher level of digitalisation of data. More digitalised data leads to higher use of data by interoperability platform.

More APIs offering access to data. The more data is open over APIs to external use the more it is used by the interoperability platform.

More digitalised government services. The more government services are digitalised the more transactions are done in the interoperability platform [GOV-RESIL].

Digital skills of employees of government IT organisation impact how much digital services are developed and how optimal the implementations are [DIG-SKILLS].

Resources for government digitalisation such as people, budget and technology affect the level of digitalisation of government organisation.

All those factors imply requirements for higher performance and better scalability due to:

- More digitalised data enables more services
- More services lead to more data exchange
- More data exchange leads to higher performance/scalability requirements
- Higher skills of government IT employees lead to more digitalised services
- More resources lead to more digitalised services

3.4. Impact on the technical interoperability solution

The analysis highlights transaction volumes, organizational complexity, and digitalization as key drivers of performance requirements. It poses additional requirements for government IT:

- Government IT systems and services must be optimised for resource usage to support the high number of transactions
- Interoperability platform UXP must be able to process a high number of transactions

In the next section will look at the abstract technical patterns that can address these concerns. In section X we dive deep into the technical scalability issues of the interoperability platform.

4. Designing the government IT systems

This section gives guidance and describes architectural patterns that can be used by government information system developers to optimise resources and make better use of the interoperability framework.

4.1. Caching information at the client side

If seldom-changing information (reference data) is used to make UI more responsive (for example, populating various dropboxes, providing autocomplete feature), it is possible to cache this information at the client IS side.

This caching strategy should be negotiated with the service provider to ensure that a) the security level of the data allows prolonged storage at the client side and b) that the client is well aware of the update policy of the data (how often it changes, is it allowed to use potentially stale cached data, etc.).

For seldom-changing reference data the service provider can design the API in a manner that enables/encourages downloading the full dataset and updating it. In this case, the API typically provides two operations: downloading a full dataset and downloading all changes

since date X. In addition, the service provider may implement push mechanism to notify the client about changes in the reference data (the push notification may contain the changes, if they are not large).

4.2. Asynchronous communication

In some instances, is possible to use asynchronous communication patterns (push instead of pull, publish-subscribe) to decrease the number of exchanged messages and possibly off-load the processing to off-peak time. Publish-subscribe can replace potentially costly polling for answers or updates.

In certain cases it may be possible to reduce the load on the interoperability framework as well on the target service by combining several atomic requests as one batch request. This approach may save on overhead by the interoperability framework (digital signatures, authentication, access control).

Some examples of situations where implementing asynchronous communication can be useful.

- When processing the request can take significant time, it is easier to send response back asynchronously instead of keeping the connection open.
- For longer processes, polling for response can be replaced with asynchronous message originated by the responder.
- If the messages are not time-critical, they can be batched together and sent in an off-peak hours. The same logic applies for sending large files.

5. Redesigning the interoperability platform to increase scalability

In addition to careful design of the government IT systems, the interoperability platform should be redesigned to allow for increased performance so that it will not become a bottleneck.

In order to cope with the large number of requests, the data exchange platform itself must also be scalable. UXP provides a good base for handling the necesary load.

- UXP uses decentralized architecture, removing the reliance on a central point and enabling direct communication between the service providers and the service clients. It enhances scalability, reduces latency, and improves reliability by avoiding bottlenecks and single points of failure. This approach also strengthens confidentiality and privacy by limiting centralized vulnerabilities.
- UXP uses the PKI services in a manner that decouples the load on the PKI service from the number of messages exchanged in the platform [HIGH-PERF], [BATCH-HASH], [HP-PROFILE].

- The load on registry server depends on the number of security servers, not on the number of exchanged messages.
- UXP has built-in load balancing with the client automatically using a round-robin algorithm if the service provider is configured with multiple security servers.

This section looks at the ways of improving the performance and scalability of UXP security server to support even bigger transaction loads. Transitioning the security server architecture from a baseline monolithic architecture to a modular architecture opens up possibilities to increase its throughput and scalability. After introducing the as-is architecture of the UXP security server, we discuss two options for increasing the performance and scalability:

- implementing elastic scaling of the load-bearing component of the security server so that new instances can be spawned in response to increased load;
- implementing modular security requirements so that security mechanisms can be relaxed for messages that are numerous but not necessarily business-critical.

5.1. Current security server architecture

The UXP security server acts as a security-enabling component on the periphery of an governmental organization. Its purpose can be summarized as follows:

- authentication of the gateway with remote peers and verification of their authenticity;
- routing of transactions between the service client and intended service provider;
- enforcing access control rules for services;
- providing non-repudiation of transactions via digital signatures;
- keeping a log of transactions and their corresponding digital signatures.

5.1.1. Security server components

Figure 4 shows the simplified architecture of the security server.

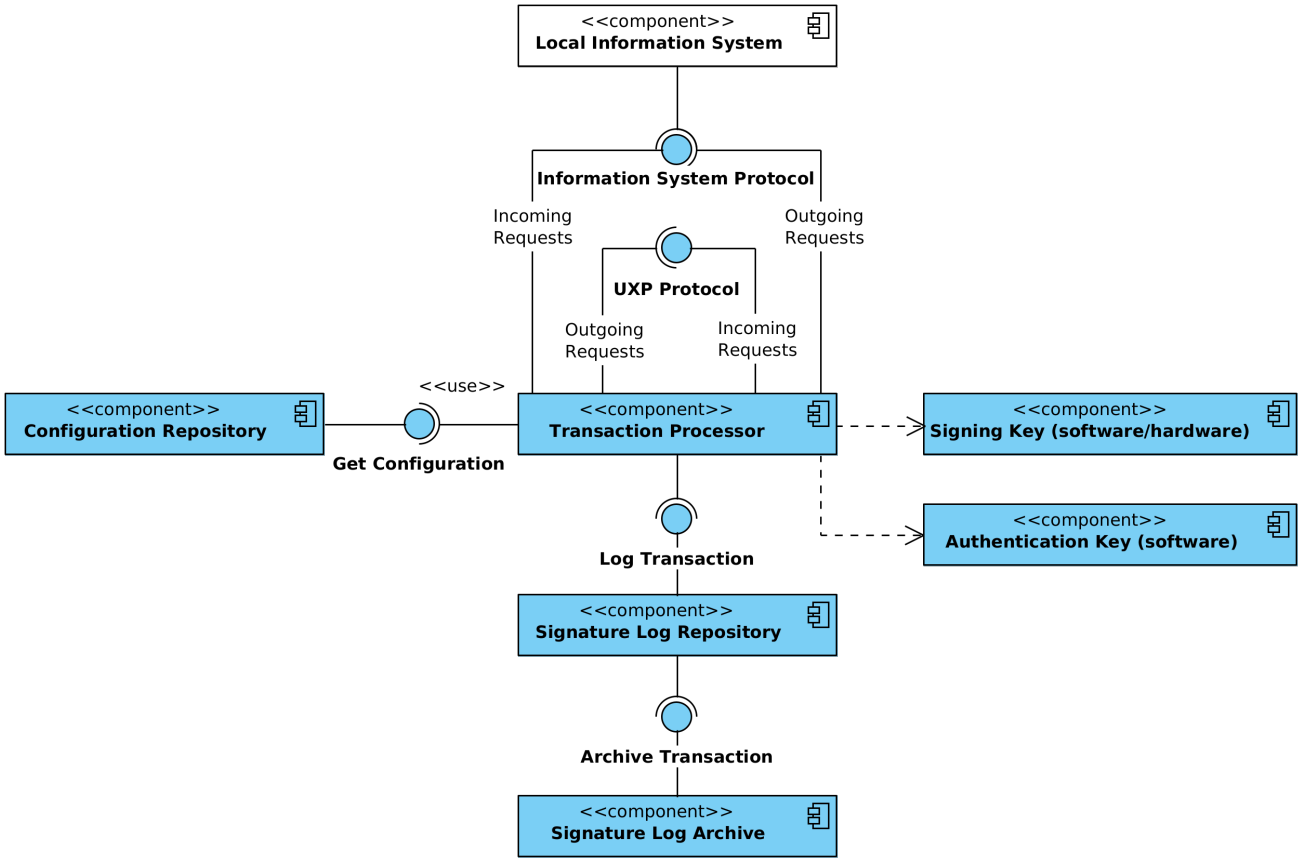

Figure 4. Security server architecture

The security server has the following components.

- Transaction processor — the load-bearing component of the architecture. Transaction processor mediates the messages between the local information system and the security server of the communication partner.
- Authentication key — used to establish mutually authenticated TLS session between the two security servers or between the security server and the information system. The authentication key is stored in security server hard disk (software key) together with the accompanying PKI certificate.
- Signing key — used to digitally sign messages passed through the security server. The signing key, togetherwith its PKI certificate may be stored either in server hard disk (software) or a secure signature creation device (hardware), depending on requirements.
- Configuration repository — contains both local configuration of the security server (access control rules, trust store and general configuration) and global configuration downloaded and cached from the Registry.
- Signature log repository — used by the transaction processor to temporarily store the signature and the message before moving them to a more permanent location. The main requirement for the signature log repository is that it must offer fast writes.
- Signature log archive — cheaper storage where the signatures and theexchanged messages are archived after timestamping.

5.1.2. Message processing in security servers

When processing a service request, the client-side security server performs the following actions (see Figure 5).

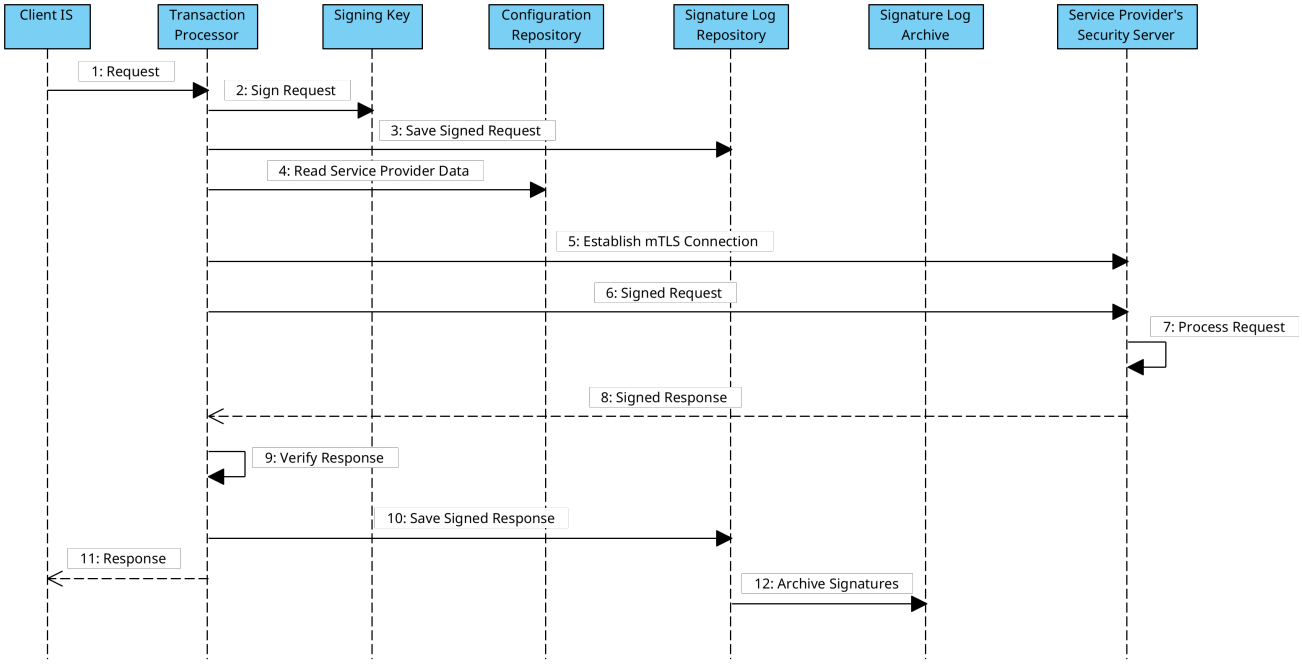

Figure 5. Message processing in client side security server

1. Receive request from the information system. The data exchange is protected by mutually authenticated TLS connection. The information system certificate is pinned in the security server configuration.
2. Sign the request using the organization's signing key.
3. Save the signed request in the signature log repository.
4. Consult the cached global configuration to find a security server that provides the service. If several security servers provide the same service, use round-robin algorithm to select a security server to make connection.
5. Establish connection to the service provider's security server. The connection uses mutually authenticated TLS, using pinned certificates, stored in the global configuration.
6. Send the signed request to the service provider's security server.
7. Service provider's security server verifies the request, forwards it to the service and signs the response received from the service.
8. Receive the signed response from the service provider's security server.
9. Verify the signature of the response.
10. Save the signed response in the signature log repository.
11. Send the response (without signature) back to the client information system.
12. Later on the signed messages are transferred to the signature log archive.

When processing a service request, the server-side security server performs the following actions (see Figure 6).

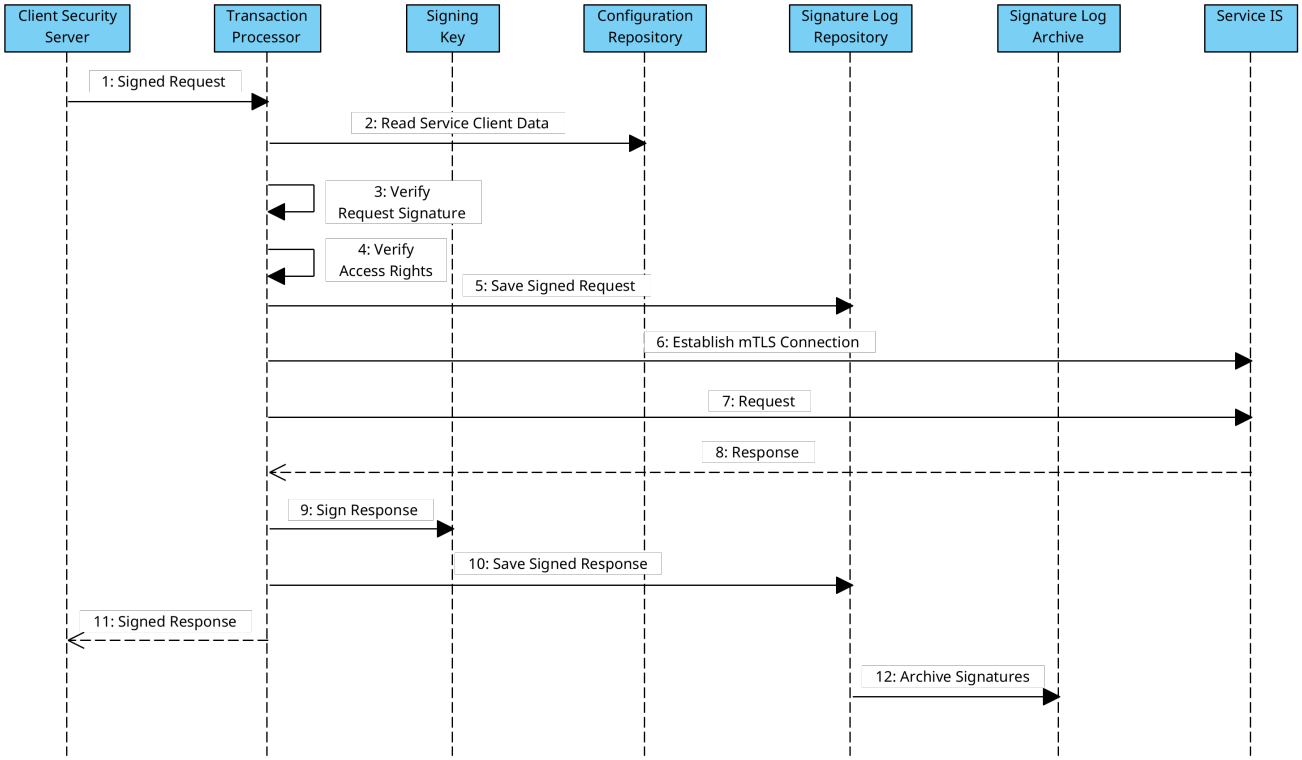

Figure 6. Message processing in client side security server

1. Receive request from client's security server. The connection uses mutually authenticated TLS, using pinned certificates, stored in the global configuration.
2. Verify the request signature and check that the request was signed by the organization indicated in the request headers. As an additional check, verify that the organization who signed the request is registered in the security server that authenticated using the TLS handshake.
3. Check the service Access Control List to verify that the client information system is authorized to access the service.
4. Save the signed request to the signature log repository.
5. Establish secure connection to the service information system. The data exchange is protected by mutually authenticated TLS connection. The information system certificate is pinned in the security server configuration.
6. Send the request (without signature) to the service information system.
7. Receive the response from the service information system.
8. Sign the response using the organization's signing key.
9. Save the signed response to the signature log repository.
10. Send the signed response to client's security server via previously established TLS connection.
11. Later on the signed messages are transferred from the signature log repository to the

signature log archive.

5.2. Elastic scaling the security servers

One of the goals of a target modular architecture is scaling the transaction processor component horizontally, by deploying a number of nodes as required by the current load profile. In the baseline monolithic architecture, all of these components are deployed on a single node (with the potential exception of a network `PKCS#11` device), which provides the architecture with implicit trust between the components — something that becomes more challenging to achieve in a modular architecture. In this subsection we discuss challenges related to dynamically scaling the transaction processor component — creating and destroying the instances in response to change in the number of requests submitted to the server.

5.2.1. Challenges of elastic scaling

Trust is the principal property of the interoperability platform — both providing trust between participating organizations and requiring trust to be established between its various components to be able to effectively fulfill its purpose. The move to an automatically scalable architecture poses challenges in both of these directions.

When it comes to providing trust between participants, one of the primary concerns in a potential elastically scaling target architecture is the authentication of the security server when establishing trusted connections for transaction processing. In the baseline monolithic architecture, the transaction processor initiates outgoing trusted connections and is the terminator for incoming trusted connections. It achieves this by loading the authentication key material into memory, a single copy of which is stored on the node in encrypted form. Making use of the encrypted authentication key material also requires credentials to be loaded into memory via administrator action.

For the following discussion we will discard the idea of cloning private keys used for authentication of the security servers (and their respective credentials) to the transaction processor nodes. This exponentially increases the impact of certain attack vectors and exposes additional ones in deployment environments that cannot be assumed to be unequivocally tamper-proof, which is generally the case. For example, if confidential key material leaks during its transfer from the configuration repository to the newly established transaction processor, the attacker can successfully impersonate such a node or execute a man-in-the-middle attack. Furthermore, the same concern applies not only to the key material, but to the credentials as well. The credentials would need to be automatically distributed to the transaction processor nodes along with the key material — otherwise elastic scaling would not be possible due to administrator action being required in response to upscaling events. These issues also exist in the case of a pre-defined pool of private keys and certificates that can be leased to individual transaction processor nodes by the configuration repository component.

The problem of security server authentication is twofold:

- authentication of the service provider’s security server by the service client’s security

server (server authentication);

- authentication of the service client's security server by the service provider's security server (client authentication).

There are two general approaches for establishing authenticated secure connections between security servers (given that options involving the transfer of existing keys to elastically created nodes were previously deemed unsuitable):

- transaction processor nodes delegate the establishment of secure connections to a separate component — a secure connection terminating proxy, which manages its own authentication key pair and credentials;
- automatically generate authentication keys and requisition domain validation certificates from a trusted certification authority via an automated protocol (such as ACME [ACME]) when a transaction processor node is created.

Let's outline the differences in the target architecture, when these approaches are applied, and consider how they play out when scrutinized in the context of the stated authentication problems.

First off, let's establish that a load balancer component is needed in the target architecture (at the service provider's side of the security server). The load balancer has two main purposes:

- to distribute incoming service requests between the elastically scaled transaction processor nodes;
- to be the sole publicly accessible network location and a known access point to interoperability platform service clients.

Thus, it is expected that transaction processor nodes do not need to be associated with external network addresses, but to be accessed through the load balancer component located in the DMZ, reducing the potential attack surface.

In the next subsections we examine the main challenges related to including elastic scaling with a load balancer: authenticating the server to the client, authenticating the client to the server, calculating the digital signatures for messages and keeping the audit log of the signed messages.

5.2.2. Server authentication

For server authentication, we see two possible technical solutions: delegating authentication to a proxy component (separate from the transaction processor) and implementing automatic certificate requisition.

When **delegating authentication to a proxy**, the load balancer (proxy) component must have its own authentication key, certificate, and credentials (provided via administrator action). In addition to decrypting requests and encrypting responses, the load balancer also has to assume the function of authenticating the service client's security server, making it quite a functionally heavy component in the architecture, as well as being a single point of failure.

Addressing the emerging high availability concerns resurfaces most of the initial issues the approach was attempting to solve. To perform its somewhat extensive functions, the load balancer will also need to receive configuration from the configuration repository component, meaning trust and a secure connection needs to be established between the load balancer proxy and the configuration repository. Additionally, establishing trust and secure connections between the load balancer proxy and the transaction processor nodes would also be a source of inefficiency. With only signature creation being the domain of transaction processor nodes, the benefit of this architectural approach from a resource utilization point of view leaves a lot to be desired.

When implementing **automatic certificate requisition**, the load balancer component must not terminate the secure connection and forward it to the target transaction processor node (TLS passthrough), which will deal with decryption, encryption, and peer authentication itself. Most of the issues that a heavy load balancer proxy creates are not present with this approach, but it relies on the availability of a trusted certification authority that supports an automated domain validation certificate management protocol (such as ACME). In this case, authentication relies on the fact that the load balancer is accessible on a specific network address, for which the domain validation certificate is issued — since the address is a registered known security server address associated with a particular organization and the domain validation certificate is issued for it, there is reasonable certainty in the server's authenticity.

In both scenarios, on the service client's side the secure connection is initiated with the load balancer's network address as the target, which is advertised as the security server's access point. The option to use automatic certificate requisition seems more promising than the alternative. However, it puts some tight constraints on the environment where the interoperability platform is deployed: the certification authority offering automated domain validation certificate management is now a hard requirement rather than a convenient option.

5.2.3. Client authentication

For the client authentication, we again look at the two possible implementation options.

In the case of **delegating authentication to a proxy**, adding a secure connection terminating proxy component to the service client's side comes with all the issues described for the server authentication scenario. The additional complexity, round trips, and trust issues bring about excessive inefficiency and vulnerability into the target architecture, which makes the ability to scale a small part of the transaction unnecessarily complicated and costly.

Automatic certificate requisition: using automatically issued domain validation certificates for client authentication of each transaction processor node is not suitable, as these certificates in general do not represent a particular organization or device, but that the peer controls a specific domain. This can be circumvented by creating the organization-to-certificate relationship within the interoperability platform configuration (UXP requires all authentication certificates to be registered for this purpose) and these associations are distributed via the global configuration to security servers. While automating the registration of domain validation certificates issued to security servers is possible, removing oversight and due diligence from the equation would open up the platform to impersonation attacks. If identification of the organization that owns the server could be made possible via the

certificate's subject information, which is unlikely for domain validation certificates, the registration could be skipped in theory and the certificates could be used as is for authentication of the client, but such a scenario is highly unlikely given the state of the art of current PKI ecosystems.

As it stands, there does not seem to be a good option for client authentication. Automatic certificate requisition is not really applicable and delegating authentication to a separate client-side proxy is in essence the separation of only the creation of advanced digital signatures away from the transaction processor to a scalable component. However, the resource utilization of this approach would be too inefficient to consider it a viable option, given the additional concerns signature calculation itself needs to contend with.

5.2.4. Signature calculation

First off, the same that has been said in regards to cloning authentication key material also applies to cloning signing key material. Furthermore, the potential regional requirement for qualified digital signatures takes the use of software keys out of the equation, since secure signature creation devices (SSCD) are required by most regional laws for qualified digital signatures. The latter aspect makes it clear that the only option for signature calculation is the use of either a networked hardware security module (HSM) or a `PKCS#11` proxy server connected to a non-networked HSM.

Since qualified certificates are usually issued on a commercial basis, signing keys need to be generated in advance and not on transaction processor node deployment. Therefore, there is little option but to have all transaction processor nodes use either a single pre-generated key and qualified certificate pair or a pool of said objects residing on the HSM that all nodes access over the network. The consequence of this is that even more responsibilities are delegated away from the transaction processor nodes and have to yet again be placed onto a chokepoint in the architecture, which becomes not only a single point of failure, but also a source of additional latency for transactions due additional data transfers and network roundtrips.

5.2.5. Distributed transaction logging

Signed transaction data and its signatures (for transactions that have legal value) are synchronously (before the transaction may proceed) written to the signature log repository and then, eventually, are moved to the signature log archive. Losing this data equates to potential loss of non-repudiation of a transaction and should be considered a security incident.

In the baseline monolithic architecture, the signature log repository is part of the monolith and provides fast write operations to the transaction processor — this property is important for the overall performance of the security server. However, there are two potential drawbacks to separating signature log repository from the transaction processor.

1. If signature log repository is on a separate node from a transaction processor, the synchronous write operation will accrue the roundtrip between the network nodes.
2. If signature log repository is shared among multiple transaction processors, it will become

a single point of failure in the system, as well as a bottleneck in terms of performance.

Given these issues, it makes sense to have each transaction processor node in the target architecture to come bundled with its own signature log repository component. However, the fact that losing a signature log should be considered a security incident, it must be ensured that the signature log contents of each transaction processor node is delivered to the signature log archive before the transaction processor node is destroyed when freeing resources during scaling procedures of the cluster.

5.2.6. Intermediate conclusion

Having conducted a thorough analysis of the potential elastically scalable target architecture for the security server, we can conclude that if we take into account all security concerns and particularities of integrating the architecture with the PKI ecosystem, we are left with a target architecture that has too much overhead for the benefit of elastically scaling a very small portion of its responsibilities.

Therefore, we need to look for alternative options to elastic horizontal scaling of transaction processors.

5.3. Relaxing the security requirements

Currently UXP aims to ensure that all the messages processed by the system will have a legal proof value. To achieve this goal, the following measures are used.

- Security servers digitally sign all the messages sent to the communication partners.
- The messages are signed using qualified certificates with keys stored in the secure signature creation devices (typically hardware security modules).
- The signatures are timestamped by a trusted timestamping service.
- The timestamped signatures are stored in a long-term signature log archive.

If the need arises, the timestamped and digitally signed message can be extracted from the signature log archive and presented as evidence in a dispute.

This security framework is aimed at business-critical transactions such as making decisions based on results of queries to government databases or for sending legal-grade documents between organizations. However, when analyzing actual data exchange patterns we find that not all messages and services require this level of security.

- UXP monitoring system uses the same UXP protocol when transferring monitoring information between organizations. These are technical messages that do not need legal-grade signatures.
- Polling for results of asynchronous operations do not need legal-grade security if they return no data.
- Health check services are often made accessible through the security server, but have no legal value. Depending on circumstances, these services can see heavier usage than the

actual operations that they relate to.

- Various queries are intended to support the user interface and thus not intended to have long-term effect. Examples include populating drop-down lists and checking user authorizations.

In all the mentioned cases the technical messages do not need the same level of protection as the business level messages.

Transaction logging is one of the most resource intensive operations that the security server performs, both in its impact on transaction processing time and in terms of storage space requirements for maintaining an archive of transaction digital signatures. Options for handling transaction logging based on service requirements could be:

- log only certain parts of the transaction data based on a heuristic (e.g., size of the logged part);
- do not log transaction data at all.

Additionally, temporarily logged transaction data could be deleted instead of being archived as part of the service processing policy.

The question arises then, why would the security server even create advanced digital signatures if they are not going to be stored as legal evidence? The digital signatures serve an important role in establishing trust within the transaction. The transaction is signed by the interoperability platform participant's key, effectively authenticating the transaction participant — especially important in multitenant deployments. However, even in this case there are options that could be provided:

- not requiring signatures at all — for cases where the participants are wholly unimportant (e.g. the health check service case);
- requiring a digital signature, but its value only, not an entire advanced digital signature;
- requiring an advanced digital signature;
- requiring a qualified advanced digital signature.

Reducing transaction processing requirements to less impactful ones for services that are not mission critical or have no legal value will significantly increase the transaction throughput of the security server in scenarios that are observed in production deployments of UXP.

5.4. Ad hoc Scaling

Automatic scaling is not the only option available to make intelligent use of available resources. In UXP, a virtual high availability cluster may be composed of a set of security servers that host a particular service provider. The word *virtual* in this case refers to the fact that the security servers in question do not need to be physically connected in any way or even have the same configuration (aside from the configuration of the set of services provided by the target service provider), but compose a cluster purely from the viewpoint of the service client. This viewpoint is built by finding from the global configuration all the security servers that host the service provider. The virtual cluster nodes remain in their

baseline monolithic and implicitly secure form.

While initially designed as a high availability solution, the fact that network addresses of all virtual cluster security servers are known to the service client's security server, means that a load balancer can be integrated into it, allowing the service client to distribute transactions among target virtual cluster security servers.

A set of configured and deployed security servers can be prepared by a interoperability platform participant organization to form a virtual cluster. Some of the security servers may be kept on standby in powered down state to save on resources and be booted up e.g. during peak demand hours. The integrated service client's load balancer is responsible to detecting when a virtual cluster security server is down and quarantine it from use for a duration, distributing load only among active nodes.

An important consideration for this approach is the ease of use of the service provider's cluster management capabilities. The number of security servers in this configuration may grow fairly large for bigger UXP participants and managing each security server's configuration independently will quickly become unmanageable, especially given the fact that a particular service provider's service set configuration needs to be kept in sync.

While the aforementioned issue can be solved by adding automatic configuration synchronization for security servers, this architectural change opens up new attack vectors. A better approach would be to solve the cluster management user experience via clever UI design when it comes to managing configuration of multiple security servers from a single unified interface. Administrator authentication, for multiple security server configuration APIs concurrently, may be facilitated by using single sign-on through OAuth 2.0 and a shared identity provider for all virtual cluster security servers.

6. Conclusion

Population, organisational complexity and level of digitalisation are the main external drivers of the number of transactions in the interoperability platform. A higher number of transactions sets requirements for better performance of government interoperability platforms. Thus, the performance of the interoperability platform is critical in service-intensive large countries.

Several performance optimisation strategies can be applied in service design and interoperability platform operations in such countries. In service design information systems can use for example client-side caching and asynchronous data exchange patterns for performance optimisations. Interoperability platform performance can be improved by differentiating the security of transactions. Non-business transactions do not usually require similar security compared to business transactions.

Two horizontal scaling options were analysed for the interoperability platform security server component. Results are mixed. In case of elastic scaling, the security considerations do not outweigh the elastic scaling benefits and are not thus recommended. For adhoc scaling the best strategy is to focus on optimization the user experience of cluster management. Automated configuration synchronisation is not advised as it opens the interoperability

platform to new attack vectors.

Further research could explore horizontal scaling strategies for nodes in mesh-type interoperability networks that rely on PKI security. Key areas of focus could include optimizing trust establishment mechanisms, ensuring efficient certificate management in case of horizontal scaling, and minimizing the impact of cryptographic operations on performance. These strategies must balance security, governance, and interoperability while prioritizing a seamless experience for end users of the interoperability platform — citizens and businesses.