

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

Gediminas Milašius

# Analysis of various multi-national eID authentication solutions in Estonia

Master's Thesis (24 ECTS)

Supervisor(s): Arnis Paršovs, MSc

Tartu 2022

# Analysis of various multi-national eID authentication solutions in Estonia

## Abstract:

In Estonia, citizens can log in to online services via eID authentication schemes such as Smart-ID, Mobile-ID, and smart cards. The vast majority of these authentications are going to banks and e-government services. If any other business wished to integrate eID authentication, they would encounter that information about authentication providers is scarce and scattered. No comprehensible resources exist that enumerate and compare various currently available eID schemes. The thesis aims to fill that gap by listing options available and providing security and integration analysis of three of them - Web eID, eeID, and Dokobit. The main finding of the thesis shows that most businesses do not need eID authentication and that most authentication providers are not mature enough to support businesses. As it turns out, the analyzed eID authentication schemes covered are more vulnerable than initially anticipated. Some were susceptible to CSRF and phishing attacks, while others suffered from lackluster and contradictory documentation. With the resources and support provided by the eID authentication companies, it would have been challenging for developers to integrate the systems, let alone harden them from various protocol attacks. The findings of this thesis will not kickstart a boom of businesses trying to integrate eID authentication. If anything, they would discourage it. However, the contributions shine a light on the problems eID providers must overcome if they wish to achieve more widespread adoption.

## Keywords:

List of keywords

## CERCS:

CERCS code and name: <https://www.etis.ee/Portal/Classifiers/Details/d3717f7b-bec8-4cd9-8ea4-c89cd56ca46e>

## Tüübituletus neljandat järku loogikavalemitele

### Lühikokkuvõte:

One or two sentences providing a basic introduction to the field, comprehensible to a scientist in any discipline.

Two to three sentences of more detailed background, comprehensible to scientists in related disciplines.

One sentence clearly stating the general problem being addressed by this particular study.

One sentence summarising the main result (with the words “here we show’’ or their equivalent).

Two or three sentences explaining what the main result reveals in direct comparison to what was thought to be the case previously, or how the main result adds to previous knowledge.

One or two sentences to put the results into a more general context.

Two or three sentences to provide a broader perspective, readily comprehensible to a scientist in any discipline, may be included in the first paragraph if the editor considers that the accessibility of the paper is significantly enhanced by their inclusion.

**Võtmesõnad:**

List of keywords

**CERCS:**

CERCS kood ja nimetus: <https://www.etis.ee/Portal/Classifiers/Details/d3717f7b-bec8-4cd9-8ea4-c89cd56ca46e>

# Contents

## **Unsolved issues**

# 1 Introduction

## 1.1 Motivation

With the emergence of COVID-19, work from home has rapidly grown in popularity. It has been especially noticeable in the IT industry. This phenomenon has led some businesses to transition to operate fully remote [?], allowing for potential customers, clients, and employees to operate with the companies' IT systems from all around the globe.

Identity verification is a significant roadblock when establishing a remote work policy. In some managerial businesses, such as logistics, it is essential to assure the authenticity of persons logging in to perform their duties. This security requirement is essential for those dealing with contracts, where one input can cost thousands. Traditionally, as work was always on-premises, it was easy to verify the identity with the help of an identity document. With the constraints imposed by fully remote operations, companies no longer have the luxury to perform such a check.

Establishing identity online for potential employees and clients is not the only use case for digital identity. Organizations such as the British Council employ privacy undermining practices. As part of the registration process for the IELTS exam, they require their customers to submit a photocopy of their identity document for verification purposes [?]. This process is a significant privacy concern since anyone could replicate the uploaded document. Having no agency over their documents is of great concern for the end-users, making them reluctant to use the company services. Replacing the document upload with a digital signature check is more secure and less privacy undermining way of performing business.

After the EU introduced the eIDAS regulation, an alternative method for identity verification became available [?]. All EU member states are mandated to implement an eID solution in their country and recognize other countries' eID solutions. Each eID solution guarantees some degree of authenticity, from substantial to high, allowing for verification of a persons' identity via trustworthy means.

Particular risks exist that businesses must be aware of before integrating an eID authentication service. There are no comprehensive resources outlining the obstacles and costs of implementing eID authentication in the private sector at this point in time. Unknown risks are an excellent deterrent to innovation, making companies reluctant to use new technologies. Proper research into this subject may lead companies to take risks associated with implementing new technology and kickstart the mainstream adoption of eIDs in the private sector.

## 1.2 Research Problem

The main goal of the thesis is to investigate what options companies have if they wish to integrate eID authentication into their day-to-day businesses and if the advantages provided by eIDs are sufficient to outweigh the risks associated with adopting new technology. From this goal, the extracted research question is as follows:

**What is the best eID authentication option available for an Estonian EU targeting enterprise for use in their Web-based Single Sign-On (SSO)?**

To help answer this question, we would need to refine it into additional sub-questions:

- What advantages do eIDs provide?
- How should an existing authentication scheme change to support eID?
- What privacy considerations must companies take when processing user data?
- What are the different eID authentication providers available to Estonia's private sector?
  - How complicated is the solution to integrate?
  - What risks does the eID provider add or take away from the consumer company?
  - What is the market reach (in countries) of a given solution?
  - Where are the weak points in the protocol used, and how should the relying company address them?
  - How expensive is the solution to operate?

With these questions answered, it would be possible for any company to understand the risks and benefits of any eID solution and have a framework to compare them.

## 1.3 Scope and goal

There will be some assumptions about the company wishing to implement eID authentication in the thesis.

**Company already uses an HTTP-based SSO (in the cloud or on-premises)** Non-HTTP-based SSOs or other esoteric authentication systems will not be considered.

**Company is committed to getting some form of eID authentication system in place**

This means market research was done, and the management found that it is favorable to invest in eID authentication, as the users would adopt this authentication mechanism.

This thesis is only interested in the technological aspect of eID integration. Any legal, political, social or any other obstacles, if covered, will only be tangential.

**The eID provider must be accessible by an Estonian company** Other countries are also capable of providing eID solutions. However, for the scope of the thesis, only solutions originating from or heavily invested in Estonia will be considered.

## 1.4 Contribution

The thesis aims to fill the research gap on the use of eID in the private sector and provide a framework for researchers or people in managerial positions to compare different eID authentication providers.

The thesis contains the following contributions:

1. enumeration of eID service providers in Estonia;
2. analysis of personal data storage under GDPR for use on authentication;
3. comparison of the different approaches eID service providers can take for integrating cross-border authentication;
4. assessment of various data transfer protocols in use for eID authentication;
5. display of example on how a company could integrate an eID service into an SSO;
6. security assessment and disclose a vulnerabilities in analyzed eID authentication providers;

## 1.5 Structure of work

The document will consist of the following main chapters:

ISO standard to pick trust level for if companies even need it

What are different options in eIDAS, e.g. what is a QESSD

What are the differences between primary services and middlemen, advantages disadvantages

What are the weakpoints in the company structure

What is the research model?

Findings about ID Card, Dokobit, and eeID

Non web-based SSO?

Ponder about the advantages of middleware pseudominization? Say instead of personal code you get some arbitrary ID that matters only in the system



## 2 Background

### 2.1 Electronic identity eID

In Estonia, digital identity has been around for over 20 years [?]. The Estonian government issues ID cards preloaded with certificates that enable cardholders to identify themselves digitally. Estonia's early adoption of eID, the political focus on digital government, has led to over 89% of internet users accessing the e-government, making it the leading country in the EU [?]. This adoption rate is in stark contrast to another EU country - Romania, where the first easy access to eIDs came in the form of new chip ID cards in August of 2021 [?]. In that country, only 16% of internet users access government services online. The adoption of eIDs in other countries lands somewhere between these two extremes.

If a company wishes to implement eID authentication in a given country, it must realize the differences between the adoption rates. For some countries' citizens, the button "log-in with eID" will instinctively make them draw their smart card or other eID capable device, whereas, for others, it would confuse and repulse them. The eID authentication method may attract or repel potential clients depending on the country, and early adopters should be aware of this adoption gap.

**eID adoption in Estonia and Lithuania** On the surface, Estonia and Lithuania have the exact eID solutions: Bank Link, ID card, Mobile-ID, and Smart-ID. However, even with the same infrastructure, we see many inconsistencies even in the case of just these two countries.

Consider Lithuania. It is possible to connect from a centralized website (<https://epaslaugos.lt>) to access the country's public services [?]. Here it is possible to authenticate via bank link, ID card, and Mobile-ID. Smart-ID is not an option. The omission of Smart-ID is strange, as most banks support authentication via the three major eID providers, which include Smart-ID.

Some banks listed on the public sector gateway portal, like PaySera, provide significant security concerns. With that bank, it is possible to access the e-government services with only email, password, and a 2FA code sent to the registered person's phone number. For similar security concerns, Estonia's Information System Authority has taken steps to deprecate bank link [?] from use in TARA, Estonia's gateway to e-government.

In Estonia, all three major authentication options, ID card, Mobile-ID, and Smart-ID, can be used to access the e-government.

### 2.2 eIDAS

The eIDAS regulation [?] provided the groundwork for recognizing the signatures issued by other EU countries by imposing strict liability and mutual-recognition requirements.

source:  
I did it  
myself  
02-27,  
have  
video  
recorded,  
will have  
to blur

The regulation introduced the concept of a Trust Service Provider (TSP), which allowed relying parties to have a trust anchor. Before, with digital signatures, it was possible to verify that the person signed documents with a certificate and it was valid. However, there was no good way of ascertaining if a trusted authority issued the signing certificate, meaning that the certificate could have been fabricated somewhere up the trust chain. Failure to validate certificate signatures could have severe consequences [?]. Establishing trust is the purpose of TSPs, a list of which is maintained by EU member states.

The eIDAS regulation requires member states to establish eID systems and integrate them into a federal plan if they haven't already. This legislation is the origin of the eIDAS node network [?]. These nodes connect across country borders, allowing users to authenticate with the eID of their home (issuer) country in the host (current residence) country. The eIDAS authentication protocol redirects the authentication requests to the appropriate country, federating the identification process. For the institutions trying to target the EU market, this provides a significant advantage since access to one node would mean access to all nodes in the EU.

The main issue private companies encounter when accessing the network is the highly exclusive access. The eIDAS network is only concerned about connecting countries. How and who can access an eIDAS node is up to the member states to decide.

**eIDAS notifications in Estonia and Lithuania** For countries to communicate through the eIDAS node network, the governments must first notify the European Commission about what eID authentication methods they will provide [?]. Other countries can then use these methods to authenticate foreign citizens into their public services.

In the case of Estonia, the country has notified the European Commission about its smart card and Mobile-ID authentication methods in 2018 [?]. Smart-ID is not a permitted method of authentication in the context of eIDAS.

In Lithuania's case, only the smart card solution is allowed - no mobile sign-in methods have been notified [?].

The governments of these two countries have revealed a gap between what they consider to be a secure and trusted source of eID domestically and what they are willing to be held liable for in the more global context within the eIDAS network.

## **2.3 eID providers in Estonia**

Applied Cyber Security Group of the University of Tartu maintains a list of e-services [?] that use at least one eID authentication method in Estonia. The following authentication methods were listed: Bank Link, ID-card, Mobile-ID, Smart-ID, TARA, and HarID.

While the list does not count Dokobit and eeID as primary eID providers, they list them as consumers of at least one of the aforementioned schemes. These services act as intermediaries, where users could indirectly authenticate with eID schemes.

### **2.3.1 Bank link**

Banks have initially created this authentication method to provide close integration with e-commerce providers to receive risk-free payments [?]. Over time it saw an additional use case - a secure and trustworthy authentication method for the public and private services [?].

Researchers found that the bank link protocol, while applicable, was "extremely insecure" [?]. From March of 2021, RIA has disabled the use of bank link to access public services [?], which accounted for only 1 percent of all authentications, revealing its overall popularity.

Due to the lack of security auditing required to satisfy eIDAS, poor market reach, and no support from the government, this authentication method will not be discussed in the scope of this thesis.

### **2.3.2 Smart card (ID card)**

ID cards are the most popular way to access their eID in Estonia, primarily due to the legal requirement of having one. The Identity Documents Act [?] states that all EU, not only Estonian, citizens residing in Estonia must hold an ID card, with which they can access public services online. Interestingly, this requirement caused the government to issue more ID cards than people are living in Estonia [?, ?]. Smart card functionality touches not only ID cards but residence permits and other government-issued cards as well [?].

There are no variable costs to allow a person to log in to websites with their smart card. For this authentication method, no per-transaction fees exist, as the certificate validity service (OCSP) [?] can be queried for free.

An end user's computer can extract an authentication certificate from their smart card with the help of special software, commonly distributed by the government agency responsible for maintaining the cards [?]. This certificate, once on the computer, can be sent to the private company's authorization server with Client Certificate TLS option [?] or with the use of a specialized helper library, using standard REST calls [?].

Qualified trust service provider for Qualified Certificates for e-signatures installs the certificates in ID cards [?], which ensures a high degree of certainty about the identity of the person authenticating.

A significant advantage of using a decentralized eID infrastructure, such as ID card authentication, is that there is no intermediary service in the operational process, allowing companies to skip going into expensive contracts with an eID service provider.

### **2.3.3 Mobile-ID**

Five years after SK ID Solutions started managing ID cards for use in Estonia, they have developed a mobile phone-friendly way to access the users' eID for use in Estonia, and

Lithuania [?]. SK achieved it by extending the functionality of phone SIM cards by adding new applications on the microchip.

The price of using Mobile-ID for the service provider varies based on usage, starting from 10 euros per month (10ct per request) to costing over 5 000 euros, where the effective cost is under 1ct for request [?]. For the end-user, mobile operators can charge an additional fee for the use of this service [?].

Accepting Mobile-ID would allow companies to access the markets of two countries: Estonia, and Lithuania, as the technical implementation is identical.

Qualified trust service provider for Qualified Certificates for e-signatures installs the certificates in a particular variety of SIM cards, capable of supporting Mobile-ID [?], which ensures a high degree of certainty about the identity of the person authenticating.

#### **2.3.4 Smart-ID**

Smart-ID is the latest and fastest-growing way of accessing citizens' eID, working in all 3 of the Baltic States [?]. The protocol utilizes mobile phones as authentication, similar to Mobile-ID. Unlike Mobile-ID, however, it does not require specialized external hardware [?]. The authentication process is handled by combining the eID server and the end user's smartphone. Despite that, it still passed the eIDAS compliance audit for the requirement of ensuring the signature private key is "with a high level of confidence under sole control" of its owner [?]. After passing the audit, Smart-ID was recognized as a QSCD, allowing it to create QES in 2018 [?].

The price of using Smart-ID for service providers, much like Mobile-ID, varies based on usage, starting from 50 euros per month (10ct per request) to over 20 000 euros, where the effective cost is under 1ct per request, based on the total amount of transactions performed within a month [?]. For users, unlike Mobile-ID [?], there are no telecommunication operators involved, and there are no additional costs.

Implementation of Smart-ID would allow users to access the markets of three countries: Estonia, Latvia, and Lithuania.

Qualified trust service provider for Qualified Certificates for e-signatures users their data centers to hold part of the private key and certificate used to authenticate users [?], which ensures a high degree of certainty about the identity of the person authenticating.

#### **2.3.5 TARA**

TARA is Estonia's primary gateway for authentication to public services [?]. TARA provides the ability for users to sign in with any of the three primary eID methods of Estonia and with the eID schemes of other EU member states. The ability to authenticate with the systems of other countries is of particular interest, as it also doubles up as the official eIDAS node of Estonia [?].

Unfortunately for private businesses, the Estonian Information System Authority intends to limit the use of TARA to public services only [?].

Technical implementation for the consumer, unlike Mobile-ID and Smart-ID, will be much easier to implement, as it uses the well-adopted protocol of OpenID Connect [?, ?].

It is worth mentioning that while the underlying authentication methods have received proper eIDAS auditing and are backed by a qualified trust service, this and all of the following authentication methods have not been audited in compliance with eIDAS or the auditing was not made public.

TARA only acts as an authentication service. It would not be able to provide means of signing documents [?]. If the business is considering expanding to allow for online digital signing, an infrastructure like TARA would unlikely be a great choice.

### **2.3.6 eeID**

Estonian Internet Foundation created the eeID service for the exclusive purpose of bringing eID authentication to the private sector [?]. It is a clone of TARA without it being Estonia's gateway for the eIDAS node network. The similarities mean that all points outlined to TARA apply to this service.

The service is new, does not have pricing tiers, and currently asks for 9ct per successful authentication request [?].

The vision of the said service is to allow users to access the markets of all EU countries. Currently, there are only fourteen countries with notified eID authentication methods [?]: Estonia, Germany, Italy, Spain, Belgium, Luxembourg, Croatia, Portugal, Latvia, Lithuania, Netherlands, Czech Republic, Slovakia, and Denmark. Currently, the cross-border authentication feature is disabled.

### **2.3.7 HarID**

Estonian Ministry of Education and Research created this service for the youth of Estonia to access different educational institutions across Estonia [?]. ID cards are only legally required to be held by citizens over the age of 15 [?], so everyone under would be unable to access their school system. HarID accepts TARA authentication methods with the addition of username & password.

This authentication method is held exclusively for the education sector and will be skipped in this thesis.

### **2.3.8 Dokobit**

In the initial list of services using eID in Estonia [?], one service stands out - Dokobit [?]. They provide services comparable to eeID in that they aggregate different eID methods in

Estonia (ID-card, Mobile-ID, and Smart-ID) and other countries. The primary difference between the two authentication providers is how they want to achieve their multi-national implementation goals. Dokobit relies on integrating each country's system individually, whereas eeID depends on using the eIDAS node infrastructure [?].

Pricing for Dokobit varies drastically, and the provided prices for the Baltic States [?] start at 50 euros per month (7.1ct per request), going down to 4.2ct per request at 500 euros per month.

Dokobit supports 11 countries: Estonia, Italy, Spain, Belgium, Latvia, Lithuania, Finland, Norway, Iceland, Poland, and Portugal [?].

UAB Dokobit is a trust service provider for Qualified validation of qualified e-signature. It means the service itself does not provide Digital Signature certificates, but eIDAS considers the results of verification of signatures trustworthy [?].

## 2.4 Authentication and QSCD

The requirements for Estonian ID cards make a clear distinction between "ADF AWP" and "ADF QSCD" applications [?]. Both software applications are loaded onto the smart card; however, only the QSCD application, guarded by PIN2, can create QES. Implication here is that for authentication with an ID card, a QSCD is not used.

The fact that smart cards do not use a QSCD for authentication does not mean that the certificate used was not audited under eIDAS. In practice, the same QTSP CA can and likely will issue both the authentication and the QSCD certificates (see figure ??). "Since the authentication certificate is issued under the same CA and the same Terms and Conditions, the certificate policy documents and other CA statements are applicable to both certificates" [?]. While eIDAS does not outline any requirements for authentication modules in the QSCDs, we will assume that the QTSP took the exact organizational and technical measures to protect the authentication key on the same physical device.

For different eID authentication schemes, companies have to assess the scheme's security on a case-by-case basis. For all of the methods analyzed in the thesis (Lithuanian and Estonian ID cards, Mobile-ID, and Smart-ID), the same QTSP signed both certificates - for authentication and signing. There seems to be no discernable benefit to issuing an authentication certificate with a different CA, so if a device offers both authentication and qualified signature creation functionality, it is highly likely that the authentication certificate is as trustworthy as the eIDAS compliant QSCD.

**Special case: third-party providers** When using an intermediary such as TARA (eeID) or Dokobit, the new eID provider acts as a new trust anchor. All rigorous eIDAS and QTSP auditing become irrelevant when using intermediary services, as systems are as secure as their weakest links. Companies must first look at the security auditing and the certifications of the intermediary; only then consider the security of authentication

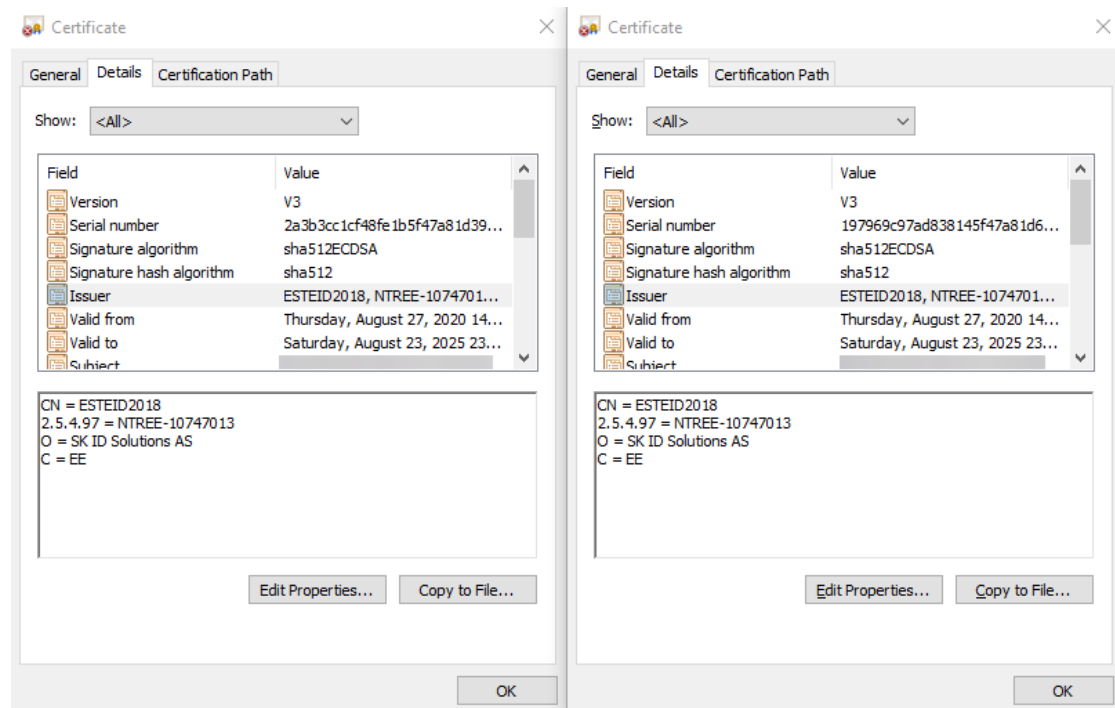


Figure 1. Comparison of authentication (left) and signing (right) certificate issuer field

options the intermediary service uses.

## 2.5 Levels of assurance

The eIDAS regulation uses three assurance levels: low, substantial, and high. These levels refer to the degree of confidence about the claimed or asserted identity of a person [?]. These assurance levels map to levels 2-4 of ISO 29115 [?].

Not all authentication methods are equal. In the simplest case, the username and password authentication method is significantly weaker than username and password with multi-factor authentication (MFA) enabled. The eIDAS framework takes this concept and builds on it in two unique ways to construct a level of assurance (LoA) [?].

The first part of determining the eIDAS assurance level is related to the registration process. The legislation distinguishes different enrollment options, such as registering on an online self-service or going to the police office to receive a physical item. The levels of assurance over the person's identity are significantly different and warrant different assurance ratings.

The second part of determining the schemes LoA is analyzing the authentication process. Username and password is less secure than username and password with MFA enabled, and both are less secure than a physical cryptographic authentication device.

The game's name here is to make the user harder to impersonate. LoA maps as one would expect - the higher the security level, the higher the level of assurance.

Companies should investigate the data they are protecting and then see what level of assurance they would need. In most use cases, the assurance level of substantial will be more than enough.

## 2.6 GDPR

When dealing with eID, sensitive personal data processing is unavoidable. Two years after the EU enacted the eIDAS legislation, the EU parliament issued new legislation to consolidate all previous privacy laws [?].

In GDPR, companies must be aware of key terminology. Interested parties can find a complete list of definitions in article 4 of the regulation. The thesis will go over only the most critical keywords and concepts.

**Personal data** The eID solutions in scope all provide the following personal information: full name, serial number (national id), and country of origin.

**Processing** The minimal amount required to process information is to store uniquely identifying information that would link an eID to an internal id code. At the very least, this action involves collecting, storing, and retrieving personal data.

**Processor and Controller** In the minimal case, no third parties are involved, and the controller, the processor, and the recipient are the same entity.

## 2.7 Threats

In the sequence diagram (see figure ??), we see a generalized high-level overview of any eID authentication solution. In the authentication protocol, the relying party (company implementing eID sign-in) entirely depends on the eID Provider, which acts as a trust anchor. A communication channel exists between the relying party and the interface, which can be categorized into two groups:

1. trusted, where the sender can reasonably guarantee that adversaries are unable to change the data in transit (Smart-ID, TARA, Dokobit);
2. untrusted, where there is a significant possibility for data to be tampered with in transit (Web eID).



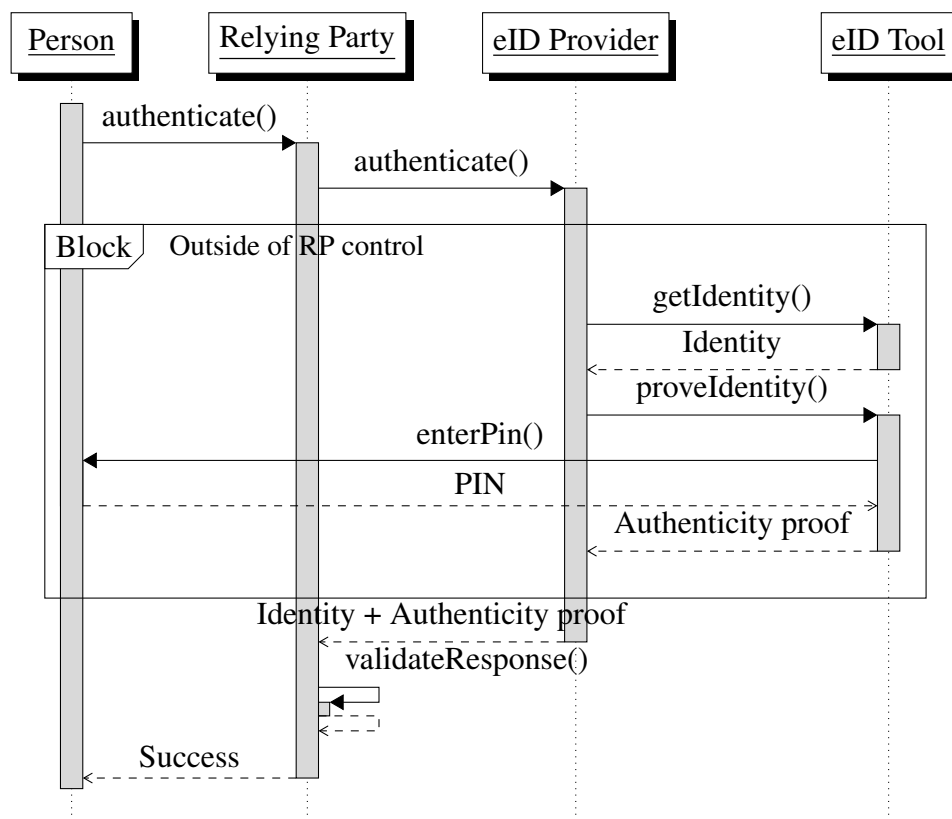


Figure 2. High-level sequence diagram of a eID authentication flow

Trusted communications commonly operate using an encrypted backchannel, whereas untrusted ones require the client to send data from the QSCD interface. Because the untrusted client sends the data, this becomes an untrusted channel, and the relying party must perform additional verification.

Failure to encrypt and establish the authenticity of the eID provider in a trusted backchannel or not validating data in an untrusted channel allows adversaries to perform man-in-the-middle attacks.

## 3 Related Work

### 3.1 National e-ID card schemes: A European overview

In 2008, researcher Siddhartha Arora investigated different uses of eID in Europe [?].

The technical report was published when the eID technology was still in its infancy, and the concept of eID was tied to it being linked to a physical ID card.

Paper references that eID cards offer three forms of information security functionality, each with an increasing level of security provisions: identification, authentication, and signature (see table ??). In this table, A is prover and B is verifier.

Table 1. Forms of information security functionality provided by eID [?, ?]

Identification (I)	A can prove to B that he is A, but someone else can not prove to B that he is A.
Authentication (A)	A can prove to B that he is A, but B can not prove to someone else that he is A.
Signature (S)	A can prove to B that he is A, but B can not prove to himself that he is A.

The idea of splitting functionality into identification, authentication, and signature can be traced to today's Estonian [?], and Lithuanian [?] ID cards. In these cards, there are two certificates — one for client authentication and the second for digital signature.

These authentication and identification certificates are not encrypted, and can anyone with the correct tools can read them from the ID card. Signed documents also have a copy of the certificate attached. These certificates identify a person, but due to ease of replication, the recipient should not trust the sender's certificate because there are no guarantees that the certificate belongs to the sender.

The authentication and signing certificates require their respective keys to perform asymmetric cryptographic operations. In theory, it is possible to sign documents with the authentication certificate; however, the verification software will reject such signatures because the certificate's purposes would not include a digital signature.

Another topic the paper touches on is the possibility of having multiple eID schemes. The author spotlights Austria as they want to have various sources of eID, not limit themselves only to one card. Having multiple authentication methods was a novel concept at the time. Many countries followed suit, and in Estonia, there are three primary sources of eID. In France, a source of eID doesn't even come from an ID card [?].

The paper's conclusion emphasizes the importance of the eID itself, not ID cards. The EU took this path when implementing the legislation for eIDAS, which allowed easier integration of infrastructure member states already had in place.

### **3.2 The Austrian eID ecosystem in the public cloud: How to obtain privacy while preserving practicality**

This paper explores what information the Austrian government stores on the issued identity documents and what operations the documents can perform [?]. Researchers identified four types of functionality: identification and authentication of Austrian citizens, qualified electronic signature creation, encryption and decryption, data storage. This functionality seems widely adopted as it matches Estonia's ID card.

Paper presented an interesting legal issue - Austria does not allow a person identifying code (CRR number) to be "used directly in e-Government applications due to legal data protection restrictions." The solution required Austria to create SourcePIN, a framework to develop different personal identifying numbers for each service trying to access it while hiding the original code [?, ?].

Authors express a big concern that everything goes through one single source of trust, which does not scale well. If many people wanted to use the system, it would quickly become a bottleneck. Moving many essential components to the public cloud can alleviate the problem.

The paper's main contribution to this thesis is to remind us that even though technological barriers are crumbling, there might still be legal obstacles to overcome. Austria is currently not part of the eIDAS node network, and it would be an excellent further research topic to investigate how Austria's eIDAS node operates.

### **3.3 Secure cross-cloud single sign-on (SSO) using eIDs**

Researchers explore the possibility of users using an SSO system to log in via their eID instead of the traditional username/password authentication method [?]. As means of doing so, they explore the capabilities of the STORK framework and other frameworks seen in previously mentioned related literature. The STORK framework is the predecessor to eIDAS [?].

The idea of the STORK framework is that any EU citizen should be able to use their eID issued by their home country to authenticate with services in other countries. An example of an activity would be opening a bank with an Italian ID card. The paper suggests extending the framework to support federation so private business identity providers can use the security options provided by eIDs and not store weak passwords.

The paper shows a proof of concept prototype usage for bringing STORK to support SSO. Emphasis was given on the backward compatibility, not to require any breaking changes to an existing STORK protocol.

Researchers found that one SAML protocol, however similar they may be, is not compatible with one another. The consumer company wishing to implement the proposed protocol must develop an adapter application to integrate different identity providers,

such as STORK, Facebook, and Google. Before a protocol sees widespread mainstream adoption, facades will be required.

### **3.4 EID @ Cloud: integración de la identificación electrónica en plataformas europeas en la nube de acuerdo con el reglamento eIDAS.**

This paper talks about integrating a new eIDAS node with the private sector in mind [?]. The eID@Cloud research initiative has proven it possible to allow private citizens to integrate this system to authenticate persons. Researchers emphasize that it does not mean ready for use and outline some issues that need addressing.

Even though the eIDAS node infrastructure brings apparent benefits to the citizens, the public, private entities, and the service vendors, there are still caveats that slow the final integration of the EU digital identity platform. The project eID@cloud shines light upon these barriers:

1. There are still some differences between the national schemes and the integrations of said national schemes in a unique and interoperable net that must be the eIDAS in the context of the EU.
2. The deployment of each eIDAS node of each member state happens at different speeds, creating mistakes and a lack of availability to complete the eIDAS project.

The interoperability testing consisted of accessing each partner's cloud platforms to verify the identities that belonged to the citizens of the other partners' countries. Norway's eIDAS node did not work with other countries' eID - the protocol executes correctly, but the user incorrectly received an error message asking for Norway identification. It shows that some parts of the system were not stable at the research time, but the whole infrastructure continued to run.

The eID@Cloud was a great project testing the implementation and readiness for public and private sectors, which provided excellent feedback for the EU Commission. The most important finding is that it found a way for private entities to connect to the mesh.

### **3.5 LEPS - Leveraging eID in the private sector**

This final research [?] was performed at a similar time to the eID@cloud [?], but in different countries. LEPS researchers have implemented an eIDAS node for private customers. However, they also provided market analysis.

The market analysis targeted four main categories of e-service providers, who would be interested in integrating eID authentication:

1. Organizations that need or want to migrate from the existing identity and access management (IAM) solution. This could apply to organizations that have scaled out their internal or tailor-made IAM solutions or organizations that already use partially external or third-party e-identification or authentication services but are looking for a higher level of assurance (LoA).
2. Organizations that use low assurance third-party eID providers such as a social login want to elevate the overall level of security and decrease identity theft and fraud by integrating eIDAS eID services.
3. Organizations that are already acting or could be acting as eID brokers.
4. Organizations that want to open new service delivery channels through mobile phone and are interested in mobile ID solutions that work across borders.

In the case of the thesis, the targeted e-service providers are of the first category - organizations wishing to improve IAM solutions to include a higher level of assurance.

The researchers recommend using an approach like LEPS to integrate eID authentication rather than creating an eIDAS node. The primary reason for avoiding node creation would be the cost-effectiveness of implementation. These adopters "are unlikely to have the know-how, resources, and capacity to implement eIDAS connectivity." "Many organizations do not have resources for eID service implementation and operation internally was already exploited by social networks." The targeted benefit is the "easy way to integrate highly scalable, yet low assurance, eID services."

LEPS is a service similar to Estonia's eeID.

### **3.6 Federated Identity Architecture of the European eID System**

The authors of this paper describe the current situation in the identity management landscape [?]. The researchers provide all the necessary background information to understand the implementation details of any eID authentication system design.

#### **3.6.1 Authentication methods**

The first significant contribution relates to explaining different ways of authenticating persons.

Any authentication method is based on something the user knows (password, pin code, answer to security question), is (biometrics - eyes, fingerprints), or has (physical device - key card, USB device) [?]. Any other method would leave the person without agency over the authentication process.

An emphasis is put on the importance of mixing and matching these authentication schemes to increase the system's security.

### **3.6.2 Authentication Paradigms and Models**

The second helpful point of the paper is the description of different identity management paradigms and models [?]. Paradigms refer to the implementation and deployment, whereas models refer to the data storage and roles.

The three main paradigms are network, service, or user-centric. The network-centric approach gathers all identities into one place, usually known as a "Domain Controller." The service-centric method would create a new identity for each service, leading to high duplication. The user-centric paradigm makes the user prove their own identity. Europe's eID solution does not favor any of the paradigms allowing identity providers to innovate [?, ?, ?].

There are also three authentication models: isolated, centralized, and federated. Unlike paradigms, where any of them is fair game, Europe's identity providers can use only the federated one. The isolated model requires all services to hold a copy of every identity in the EU. The centralized model is suitable for having a central place for looking up identity in a country, and it is an excellent solution for high-profile agencies. The federated system can scale well horizontally (adding more servers would increase the network capacity) and not require keeping an index of all citizens it would like to serve, which is a tremendous advantage when considering the GDPR requirements.

### **3.6.3 Authentication protocols and services**

Researchers have allocated a good portion of the paper to provide an overview of potential protocols and implementations. The list is massive and in-depth; however, it becomes clear that SAML [?], OAuth2.0 [?], and OpenID Connect [?] protocols are by far the most popular protocols to choose for implementation. The engineers behind the eIDAS network implementation decided to settle on the SAML protocol.

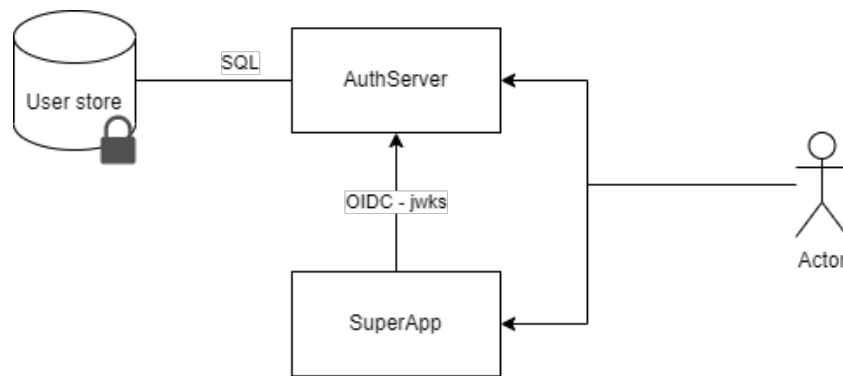


Figure 3. Initial system architecture

## 4 Architecture Definition

Redo the validation sections into 1) security requirements, 2) security requirement compliance by eID provider, 3) security requirement compliance by developers

### 4.1 System architecture

This section will look at our company's systems' current and desired states before and after integrating an eID provider. We will also declare common terminology used in the case studies, referring to the same server with the same name everywhere.

For namesake's clarity, the company wishing to integrate eID authentication is a physical exercise data tracking enterprise, **WorkAuth**.

#### 4.1.1 System overview

**Initial state** In figure ?? we see a high-level overview of a system where we are trying to integrate eID authentication. This system consists of the following components:

1. AuthServer. The company's SSO. Acts as a central authority for identity. This server issues OIDC id tokens containing user ID numbers, roles, and claims. It is the primary access control mechanism.
2. SuperApp. A resource server with access control enabled. It uses OIDC id tokens issued by AuthServer and verifies them using cryptography or other means. It contains the data the company is trying to protect with stricter access control.
3. User store. A data store containing user log-in information - usernames, password hashes, other Personally Identifiable Information (PII).



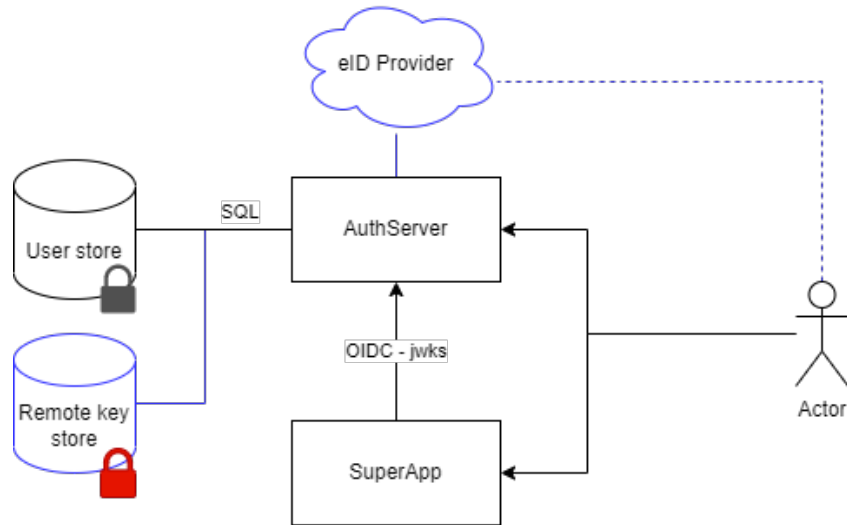


Figure 4. System architecture after the inclusion of an eID provider

4. Actor. A person accessing the resources in the system. Typically can be anyone, including computers; however, in our thesis's scope, we will limit ourselves to only natural persons holding an eID.

**Desired state** The company WorkAuth wishes to implement eID authentication. Since the authentication is not done locally but is delegated to some remote service or device, the protocol can be treated as an external federated log-in. Application development frameworks such as ASP.NET Identity have helpful tools to handle external identity providers.

With the inclusion of an external eID provider, we can compose the new system architecture to be akin to figure ???. In this figure we can see two significant additions:

1. eID provider. It is WorkAuth's gateway to obtain someone's eID. It can be any eID source such as Dokobit, TARA, Smart-ID, or an ID card.
2. Remote key store. Storage for unique identifiers provided by the eID provider.

The primary purpose of the remote key store is to link the user ID used in the internal system with the unique identifier given by the eID provider. Because the unique identifier can change or the same physical person can have multiple eIDs [?], companies should foresee cases where numerous eIDs could map to a single internal ID.

In the diagram (figure ??), the key store has a red lock next to its icon. Depending on the country, the data stored there may be subject to strict privacy regulations. Companies must consider implementing strict access control for this part of the infrastructure.

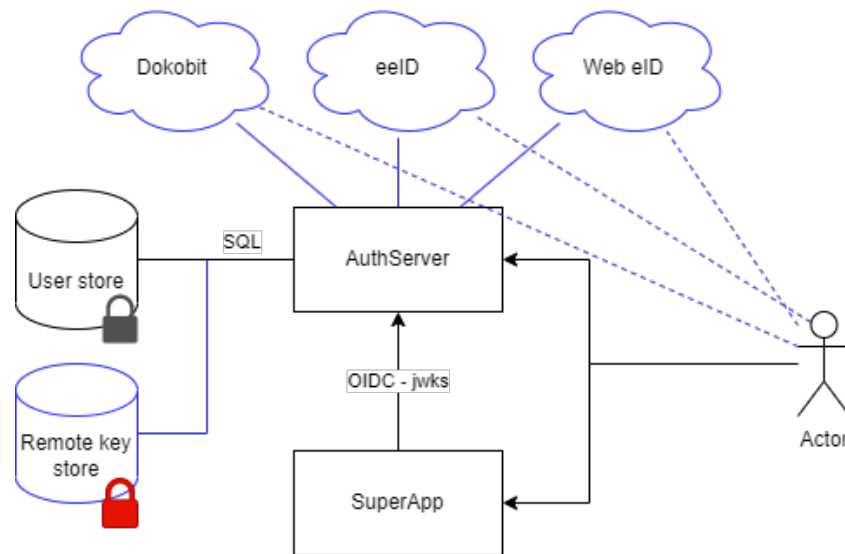


Figure 5. System architecture after the inclusion of all eID providers in scope

**Final state** The end goal for the scope of this thesis is to implement three eID providers into the architecture. For regular companies, it would make sense to implement multiple only if they would like to get more coverage. Additionally, they could register non eID providers, such as Google or Microsoft social log-ins; however, we will not cover this in the thesis. The final high-level overview of the system can be seen in figure ??.

#### 4.1.2 Process overview

For validation of the architecture, we will consider two use cases.

The first one (see figure ??) is concerned about accessing a protected resource with a token issued by the AuthServer. This use case validates the base state of the system.

The second use case (see figure ??) is also concerned about accessing a protected resource, but only those, who authenticate with a higher level of assurance, like an eID solution, would be able to access it. This use case validates the successful implementation of eID authentication and access control.

#### 4.1.3 Linking eID to an internal user ID

Possible mess? Suggestions wanted.

There will be a need to uniquely link an identity to an internal account in the company SSO server. The security requirement, in this case, is not to allow other users to access the same account. For this goal, companies must use one or more person-identifying properties.

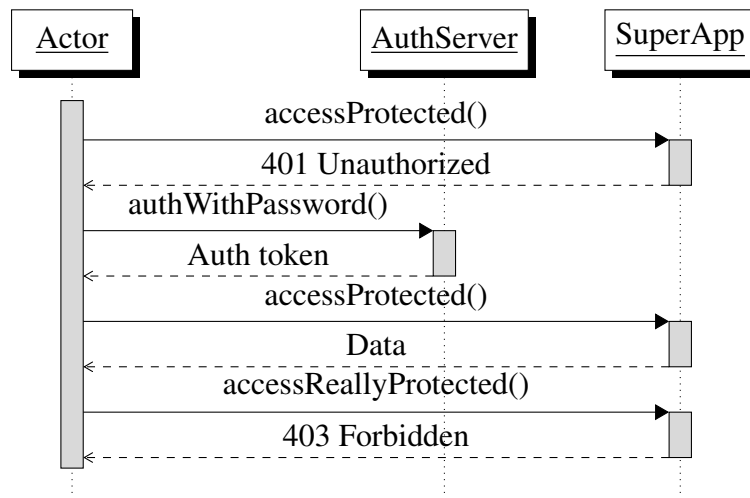


Figure 6. System behavior when authenticated with a username + password scheme

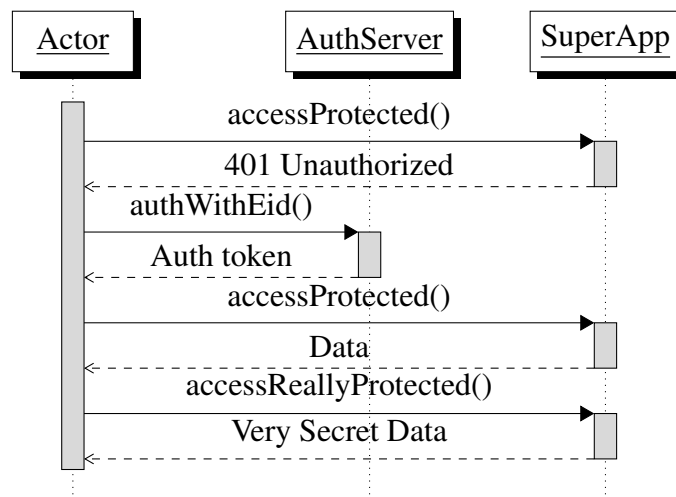


Figure 7. System behavior when authenticated with an eID scheme

Using Estonia's passport as a reference, we see that it has the following identifiers: (issuer) country code, document number, surname, given name, personal code, citizenship, date of birth, document date of issue, expiry, and authority. In these cases, it is easy to use the personal code for identifying a person as it is unlikely to change - people change names, documents expire. Document authority information and date of birth do not narrow it down nearly enough. The use of a personal number seems like an obvious choice.

Unfortunately, not all countries have personal numbers. If we look at Ireland's passport, we would be unable to find such a code. The lack of an id number shows one of the many challenges eID authentication adopters will face. If the data on the passport is all we would have, the next best unique identifier for linking users would be to use the document number. It is cumbersome and would require users to update the account with a new number when the document eventually expires and is replaced. Still, no other option satisfies the security condition requiring an eID to link only to the correct user as effectively as this one.

Fortunately, there is hope to avoid ambiguity in the digital world. European Telecommunications Standards Institute has identified the issue of countries having different people accounting systems and proposed a prefix system to account for it [?]. In their system, each ID code consists of a three-character identity type reference, followed by a two-character country code, a dash (-), and then the code unique in the country. Identity references include passport numbers (PAS), personal numbers (PNO), tax references (TIN), or even custom ones (XX:). Example of Estonian identity number under this system - PNOEE-30303039914.

If we look at the eIDAS node network, its architecture does not have strong opinions on how an ID should look, as long as it has a country code and a unique code inside the country. How countries achieve uniqueness is not of concern.

The eIDAS node architecture solves another issue of what to do if a person's identifier changes. A unique identifier remains "unchanged for the lifetime of the account" [?]. If an identifier were to change when "the user's digital identity is replaced or repaired," relying parties should treat the newly obtained identifier as a completely new identity.

**eIDAS Unique Identifier Structure** In eIDAS SAML Attribute Profile, an identifier code is defined as:

1. The first part is the Nationality Code of the identifier. Value is an ISO 3166-1 alpha-2 code, followed by a slash ("/").
2. The second part is the Nationality Code of the destination country or international organization. Value is an ISO 3166-1 alpha-2 code, followed by a slash ("/").
3. The third part is a combination of readable characters. This sequence of characters uniquely identifies the identity asserted in the country of origin but does not nec-

essarily reveal any discernible correspondence with the subject's actual identifier (for example, username, fiscal number, etc.).

Example: ES/AT/02635542Y (Spanish eID Number for an Austrian SP).

**Summary** Using the eIDAS Unique Identifier structure as a base, we can see that it is enough to uniquely identify a digital identity in eIDAS with the country of origin, country of destination, and a set of characters to identify that person in the origin country. The destination country will always be the same in our company's case. To uniquely identify a person, we will only need its origin country and a unique identifier a member state must provide.

For this thesis, identifiers will be marked as "{ISO 3166-1 alpha-2}/{Code provided by country}". Unique identifier examples: EE/38001085718, LT/49003111045, SE/870314-2391.

#### 4.1.4 Privacy Policy

A company wishing to implement eID authentication will have to deal with personal information as described by GDPR [?]. Before going live with an eID solution, they must conduct a legal audit for compliance.

A privacy policy is a legal document and is way outside of the scope of a technical implementation thesis. However, it is still important to understand the basics. For this goal, two privacy policies will be analyzed: Web eID [?] and Dokobit [?].

From the cursory analysis of the two policies, there are three fundamental aspects our company needs to address: what data is processed, with whom the company shares the data, and what is the retention policy. An actual privacy policy varies drastically from case to case.

Based on the privacy policies of Web eID and Dokobit, we constructed a rudimentary privacy policy for the use of the test application environment. The thesis appendix contains a copy of the text.

## 4.2 Weaknesses in the architecture

When analyzing the weaknesses of WorkAuth's internal systems, it is good to understand their cause. Some vulnerabilities can appear due to existing issues, faulty implementation, or deeply rooted issues in the architecture itself. In this chapter, we will address these categories.

**eID provider dependant weaknesses** These threats come from one common issue: trusting an eID service provider when a relying party should not have. There are four primary causes:

1. relying party trusts an input from an insecure channel;
2. relying party does not perform the necessary validation mandated by the protocol;
3. inherent weakness in the architecture of the used transport protocol;
4. eID provider itself gets compromised;

The thesis will address these points on a case-by-case basis for each eID provider in the study. We will address each of these points in the case studies' respective chapters.

**Local weaknesses** In figure ??, four system components are not linked to a particular eID: AuthServer, SuperApp, application store, user store, and remote key store. We can identify weaknesses in those parts in this section. We will analyze each weakness through the lens of CIA security analysis.

#### **4.2.1 SuperApp**

**[CI] Users can see and or edit data normally forbidden to access** This issue is created by one of the following:

1. access control measures are disabled on a specific resource;
2. OIDC token validation is disabled or misconfigured;

The first cause has a trivial fix - developers have likely forgotten to enable the data access guard on the endpoint. It is unlikely that all authorization rules break.

In the case of the second cause, some corners were likely to be cut in the ID token validation process. When validating a token, the process has to match the one described in the OIDC spec [?] exactly. This process consists of three major parts:

1. check if the token's crypto algorithm is as expected;
2. validate token signature;
3. validate claims - issuer, audience, timestamp, nonce;

Developers usually need not worry about the process, as most frameworks have adopted the OpenID Connect protocol or have well-maintained libraries.

**[A] Service becomes unavailable** This threat is caused by one of the following:

1. server is offline or overloaded;
2. OIDC token validation is misconfigured to have an incorrect authority;

The first case is a common availability issue, meaning the server is suffering a denial of service attack. We will not cover the mitigation of this form of availability threat in the scope of the thesis.

A likely cause for the second part of this issue is the manually configured OIDC properties on the relying party. If possible, developers should never configure the properties manually and use the well-known metadata endpoint instead. A metadata endpoint usually looks like this - `https://auth.mycompany.org/.well-known/openid-configuration`.

#### 4.2.2 AuthServer

All of the points that apply to SuperApp also apply to AuthServer. However, there is a critical use case that is worth mentioning explicitly.

**[CI] Users can see used eID schemes and add new ones with unsafe log-in** As per usability requirements, users must be able to assign multiple identity providers to their accounts. When adding a new external scheme, the currently logged-in user must have the same privilege level as the scheme they are trying to add. This countermeasure is in place if an adversary gains access to their account; they wouldn't be able to elevate their privileges by adding their eID scheme and signing in with it.

With this requirement in place, registration with an email and password becomes less valuable, as those accounts could never add an eID afterward. Companies can enact an exception to this security requirement for the first eID scheme a user would add. However, a better solution would be to have a company policy to verify the user's first added eID manually.

#### 4.2.3 Data stores

All three data stores have the exact issues between them. For the sake of brevity, we will group them under the umbrella of *data store*.

**[CI] Users have a less secure way of accessing the data store** The system is as secure as its weakest link. If users or developers have direct access to the database while bypassing the eID authentication check, the security and assurance guarantees are worthless.

If there are alternative ways of accessing the database, companies must implement proper access rules that would be as secure as the one's eIDs provide. How companies can achieve that depends on what data storage options they use. The best option would be to completely close down external or internal access to the data stores.

**[CI] Man-in-the-middle attacks** While uncommon, some data stores are vulnerable to MitM attacks [?]. Although the best course of action would be to move the data storage server away from the internet and make it accessible only on the local network or via a VPN. However, in cases where an attacker has access to the internal network, even that is not enough. For maximum security, companies must implement the recommended MitM prevention techniques [?].

**[A] Data is destroyed or lost** Ransomware attacks and accidental database corruptions can happen, so offline remote backups are a must.

An interesting issue arises when companies should treat backups to the same security standards as live databases. One approach to ensuring data integrity and confidentiality would be encryption; keys should only be available after authenticating with an eID scheme. One equally secure alternative would be to use something like the encryption and decryption functionality of Estonia's ID cards.



## 5 Case Study: eeID

### 5.1 About

The Estonian Information System Authority has created TARA - a gateway for public sector services to integrate eID authentication easier and cheaper [?]. It is heavily inspired by the OpenID Connect [?] protocol to communicate between the service and relying parties.

The Estonian Internet Foundation has then created the eeID service - a clone of TARA, but with the intent to open it up for private businesses at a premium [?].

These services' goal is to use domestic eID providers (ID cards, Mobile-ID, and Smart-ID) and act as a gateway to the eIDAS node network.

Cross-border authentication supported by these schemes extends (or will soon extend) to the notified countries [?]. Currently notified countries (in order of time notified) include Germany, Italy, Croatia, Estonia, Spain, Luxembourg, Belgium, Portugal, Czechia, Netherlands, Slovakia, Latvia, Denmark, Lithuania, Malta, France, and Sweden.

At the time of writing, TARA does not support the eID schemes of the last three - Malta, France, and Sweden, and cross-border authentication is only in the planned state for the eeID service and is currently not supported.

### 5.2 Data Flow

The underlying data transfer protocol used by eeID is almost identical to OpenID Connect code flow [?, ?]. One irregularity exists between the OIDC spec and TARA, which makes the whole flow non-compliant to the spec [?]. Otherwise, the flow is identical. A high-level overview of this flow can be seen in figure ?? . It consists of seven main steps:

1. Initial log-in. Preparation for redirect to the eID provider.
2. Redirection to the eID provider.
3. Authentication. Users log in with the chosen authentication scheme (ID card, Mobile-ID, Smart-ID, eIDAS).
4. Redirection back to the Auth server.
5. Browsing session verification. Cryptographically check if the request came from the same browser. This step is required to protect against CSRF attacks.
6. Token acquisition. The Auth server exchanges the received code for an identity token containing the user's information.
7. Signature verification. Verify the authenticity and validity of the received token.

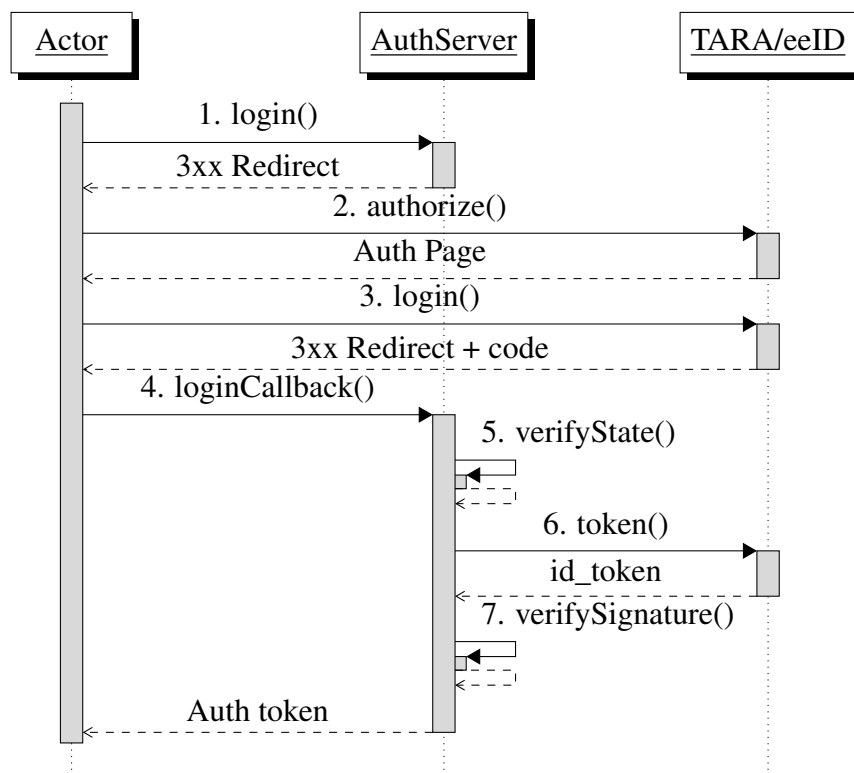


Figure 8. OIDC code flow used in TARA/eeID

### 5.3 Trust Anchor

One of the main advantages of using eeID is the simplicity of integrating multiple sources of eID (ID card, Mobile-ID, Smart-ID, and eIDAS) with a single API. To accommodate this, eeID acts as an intermediary service, reading and reprocessing data to create a normalized response. With this approach, there are two issues: confidentiality and integrity.

Confidentiality issues occur whenever users do not wish to share their personal information with more parties than absolutely required. For example, if a user wants to authenticate with Smart-ID, only two parties should be involved: the company user is trying to log in to (WorkAuth) and the Smart-ID service provider (SK ID Solutions). With the eeID service, another entity is added between them who has full read and write to the data.

Integrity issues are caused by having that full access to data. Should the eeID service get compromised, attackers could impersonate anyone in all services relying on eeID.

Companies should keep these risks in mind when integrating the eeID identity provider.

### 5.4 Pricing

The operational cost when using the eeID identity provider is nine cents per successful authentication [?]. It is the highest price among all identity providers and does not have tiers. A thousand successful authentications would cost WorkAuth 90 €.

### 5.5 Security Requirements

The TARA documentation has an in-depth integration guide and the validation requirements clients must implement [?]. Additionally, RIA claims that the protocol used is mostly OpenID Connect compliant. Because of this, we can have two resources to validate the resilience of the architecture - TARA docs themselves [?] and IETF's OAuth 2.0 Security Best Current Practice document [?].

**Communication channel** The eeID service uses a secure communication channel, encrypted end-to-end using HTTPS. When the system is integrated correctly, malicious clients (or user agents) have no possible way of influencing any of the authentication parameters without triggering alarms when validating the request.

#### 5.5.1 Protocol's built-in security features

OpenID Connect specifies three different flows - code, implicit, and hybrid [?]. In code flow, all sensitive tokens are handled via backchannel - a secure communications channel

where the user client is not involved. The implicit flow is the opposite - a client receives an identity token and sends it to the company's authentication server, which validates the signature. A hybrid flow is a mixture of the two. Security experts consider the code flow to be the safest option of the three [?]. Coincidentally, it is the only supported flow by TARA and eeID [?].

### **Replay attacks**

- eeID will reject the second POST /token request with the same code.

The developer does not have to implement internal state management to verify that a given session token was used only once. When combined with other countermeasures available, a replay attack becomes impossible to execute by any practical means. Mitigation measures provided by the eID provider are sufficient.

Image  
proof?

### **Insufficient Redirect URI Validation**

- Changes in the registered OIDC application undergo manual verification by eeID employees and do not allow for wildcards.

The manual verification process is sufficient to mitigate this attack; it is impossible to test if countermeasures exist on the client registration interface used by the eeID service employees. Mitigation measures provided by the eID provider are adequate.

### **Credential Leakage via Referrer Headers**

- eeID does not include third-party resources (javascript, image, or other); therefore, it cannot leak any query parameters to third parties.
- The company is required not to have any third-party resources on the authentication and redirect pages.

The eeID service does not leak credentials to third parties anywhere on their log-in page (all resources used in their service are from their domain, see figure ??). No mitigation measures are required.

### **Credential Leakage via Browser History**

- Replay attacks are mitigated against, the only form of attack possible with this.
- The form post response mode is not supported. Not required, but it would prevent this form of attack completely.

Attackers can obtain authorization code, nonce, and state from the browser history (see figure ??); however, for them to be able to use codes, the relying party must be susceptible to CSRF attacks. Mitigation measures are sufficient only when CSRF mitigation is in place.

Name	Status	Domain	Type	Initiator
ExternalLogin?returnUrl=%2F	302	auth.eid.gedas.dev	document ...	Other
authorize?client_id=oidc-b0669946-896...	302	test-auth.eeid.ee	document ...	auth.eid.gedas.dev/l...
login?service=https%3A%2F%2Ftest-aut...	200	test-auth.eeid.ee	document	test-auth.eeid.ee/oj...
main.css	200	test-auth.eeid.ee	stylesheet	login?service=https...
eis_logo_eng_rgb_horizontal.png	200	test-auth.eeid.ee	png	login?service=https...
main.js	200	test-auth.eeid.ee	script	login?service=https...
cef-logo-en.svg	200	test-auth.eeid.ee	svg+xml	login?service=https...
roboto-regular-webfont.woff2	200	test-auth.eeid.ee	font	test-auth.eeid.ee/stv...
roboto-medium-webfont.woff2	200	test-auth.eeid.ee	font	test-auth.eeid.ee/stv...
roboto-bold-webfont.woff2	200	test-auth.eeid.ee	font	test-auth.eeid.ee/stv...
login?service=https%3A%2F%2Ftest-aut...	200	test-auth.eeid.ee	document	Other
main.css	200	test-auth.eeid.ee	stylesheet	login?service=https...
eis_logo_eng_rgb_horizontal.png	200	test-auth.eeid.ee	png	login?service=https...
main.js	200	test-auth.eeid.ee	script	login?service=https...
form-check.js	200	test-auth.eeid.ee	script	login?service=https...
roboto-regular-webfont.woff2	200	test-auth.eeid.ee	font	test-auth.eeid.ee/stv...
roboto-medium-webfont.woff2	200	test-auth.eeid.ee	font	test-auth.eeid.ee/stv...
roboto-bold-webfont.woff2	200	test-auth.eeid.ee	font	test-auth.eeid.ee/stv...
cef-logo-en.svg	200	test-auth.eeid.ee	svg+xml	login?service=https...
login?service=https%3A%2F%2Ftest-aut...	302	test-auth.eeid.ee	document ...	Other
callbackAuthorize?client_id=oidc-b0669...	302	test-auth.eeid.ee	document ...	login?service=https...
authorize?client_id=oidc-b0669946-896...	302	test-auth.eeid.ee	document ...	test-auth.eeid.ee/oa...
signin-tara?code=OC-131-rZmyzx2qMU...	(failed)	auth.eid.gedas.dev	document	test-auth.eeid.ee/oj...

Figure 9. The eeID service does not use resources outside of their domain in the authentication flow

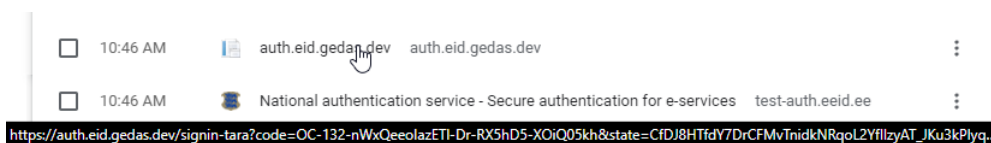


Figure 10. Authorization code is leaked inside the browser history when using eeID authentication

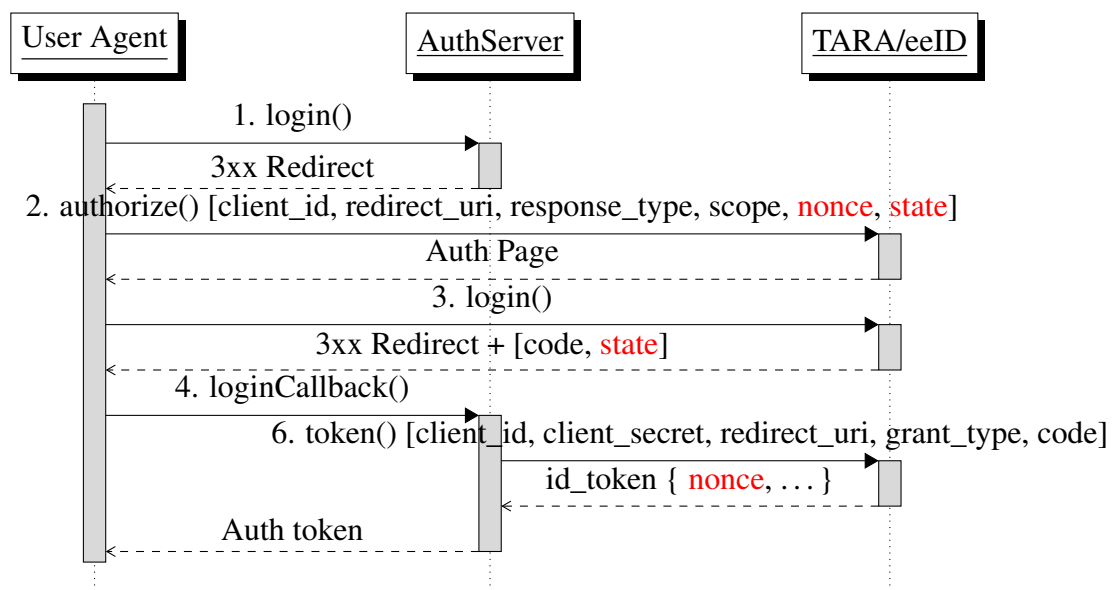


Figure 11. CSRF/Code injection mitigation in OIDC code flow

## Authorization Code Injection and Cross-Site Request Forgery

- The use of state parameter prevents CSRF, but not code injection attacks. Because the state is not bound to an authorization code, an attacker can perform a high-tech attack (such as MitM) and inject their code into someone else's user agent or steal someone else's code.
- The eeID service optionally supports the OIDC nonce parameter, fixing the injection attack. When a company redirects the user to the authorize endpoint, a nonce parameter will be bound to the given code response. After the relying party receives the identity token, they can check if the nonce parameter matches the one sent initially. If they are not, an attacker likely has injected a session token.

Figure ?? illustrates this mitigation well. Because an attacker does not have access to both authentication requests (1 and 4), they would be unable to influence either nonce or state. If the relying party validates the integrity of both nonce and state, mitigation measures are sufficient to protect from both CSRF and code injection attacks.

## Clickjacking

- The eeID service's auth page does not use Content-Security-Policy, however it does use header X-Frame-Option: DENY (see figure ??).
- Relying party should integrate this or a similar countermeasure.

▼ Response Headers View parsed

```
HTTP/1.1 200 200
Date: Sat, 02 Apr 2022 07:45:59 GMT
Server: Apache/2.4.38 (Debian)
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
Strict-Transport-Security: max-age=15768000 ; includeSubDomains
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-XSS-Protection: 1; mode=block
X-Application-Context: cas:standalone
Content-Language: en
Vary: Accept-Encoding
Content-Encoding: gzip
Keep-Alive: timeout=5, max=99
Connection: Keep-Alive
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
```

Figure 12. Response headers browsers receive when opening the eeID service's login page

Mitigation measures are sufficient on almost all browsers released in the last ten years [?].

### 5.5.2 Validation requirements for the relying party

In the previous section, we saw the security features of the protocol. The relying party should implement validation and reject requests for some of the features if those validations do not match. This section will contain all of the features the RP needs to address.

## Misconfiguration Attacks

**Incorrect OpenID Configuration** The relying party should make sure to use the correct OpenID Connect discovery document. For test environment the URL is `https://test-auth.eeid.ee/oidc/.well-known/openid-configuration`, and for production - `https://auth.eeid.ee/oidc/.well-known/openid-configuration`. A connection with a malicious party could be established when using the incorrect URL.

Developers should make sure these values are not easily editable (such as placed in environment variables) by anyone. Best they should be hardcoded in the application.

**Weak TLS configuration** TLS is the primary defense mechanism against MitM attacks when connecting to the eeID servers. WorkAuth's IT Ops team should ensure

that the server does not trust any malicious CA.

**Authorization Code Injection and Cross-Site Request Forgery** The primary validation user has to implement is protection against CSRF and code injection. Developers can accomplish this by using the state and nonce parameters of the protocol.

**Attack description** For reference to how the mitigation works, we will use the figure ???. We will provide three examples for the countermeasure: no state and nonce, strong state and nonce, and strong state and nonce. Because eeID and TARA both require sending the state parameter, for this example, we will use a weak constant state string, such as "x".

Our attack exists on the third step of the OIDC code flow - right after the user agent receives the redirect response but before they are redirected. An attacker can stop a request by changing the victim's firewall settings to block the WorkAuth servers and would be able to extract the authorization code from the browser history. This attack, however, is just a single example of countless similarly creative attacks.

**Attack #1 - No state and no nonce** After the user agent gets redirected, the attacker can extract both code and state. If no session binding is performed, they can use the URL themselves and authenticate with the victim's identity.

**Attack #2 - Strong state without nonce** This attack is similar to the first example but with the introduction of user agent session binding. The relying party, before they redirect the user agent (step 1 response in figure ??), would issue a cookie on who's value the state parameter depends. An example would be to have a session token, a regular 32-byte random string, and the state parameter would be an SHA-256 hash of that string. Attackers would not be able to reverse the hash to fabricate the cookie, and if the server, when validating the request, notices that the hash of the session cookie does not match the state, it would reject the HTTP request.

A way to circumvent this for attackers would be to use two sessions. Attackers would create a session with the relying party and wait for a victim. The victim creates a session, gets redirected, and authenticates. After the victim tries to return, the attacker stops them, takes only their authorization code, and puts it in their request. When the relying party validates the request, it would see that the attacker's cookie's hash matches the attacker's state parameter.

**Attack #3 - Strong state with nonce** To protect against authorization token injection, the relying party must bind the user agent session to the state parameter and the identity token (the result of consuming the authorization code). The relying party can achieve this by using the nonce parameter.



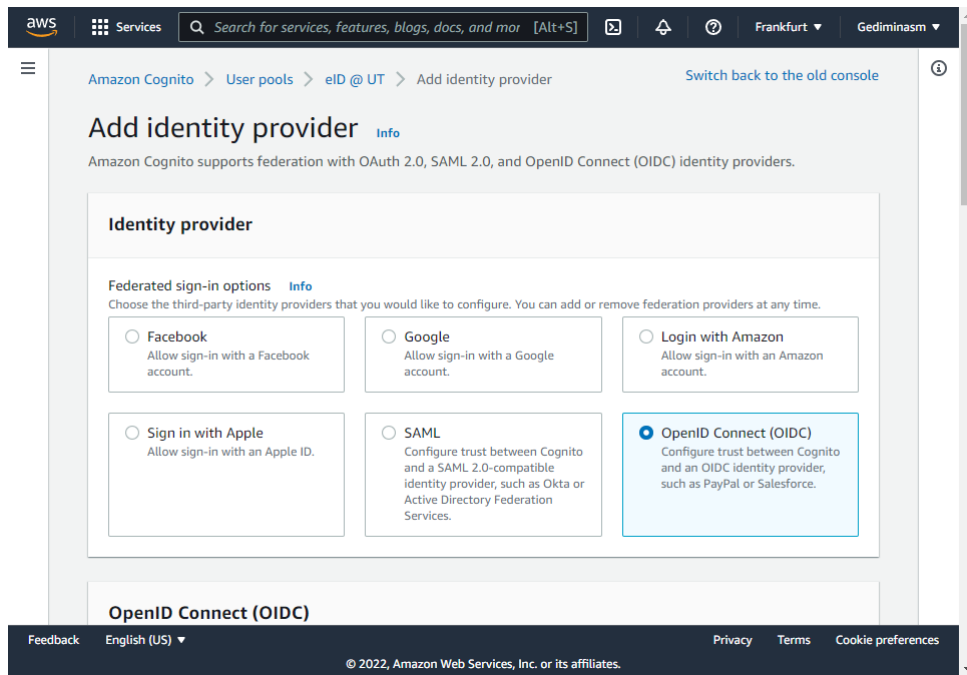


Figure 13. Adding OpenID Connect provider to Amazon Web Services

In step 6 of figure ??, we see that if we use the nonce parameter, additionally, we receive a nonce value in the identity token. This value exactly matches the one sent in the initial authorize request (step 2). If we store nonce and state inside of a user agent cookie, we can see that this prevents the previous attack, as it required victims to create a session themselves, and by doing so, they would generate a nonce that would be different than the attackers. Unlike the state parameter, they do not know what nonce was used and are unlikely to guess.

If the attackers manage to guess the nonce, they would also be required to update their user agent cookie. For relying parties to protect against this attack, they can encrypt this cookie to prevent external tampering.

## 5.6 Integration

The best way to integrate the eeID service would be to use existing OpenID Connect implementation options. Cloud hosting giants Amazon Web Services and Microsoft Azure offer an easy way to incorporate an OIDC provider (see figures ??, ??). Alternatively, there are many officially certified services and libraries developers can use [?].

This thesis will use Microsoft's official ASP.NET Core library [?].

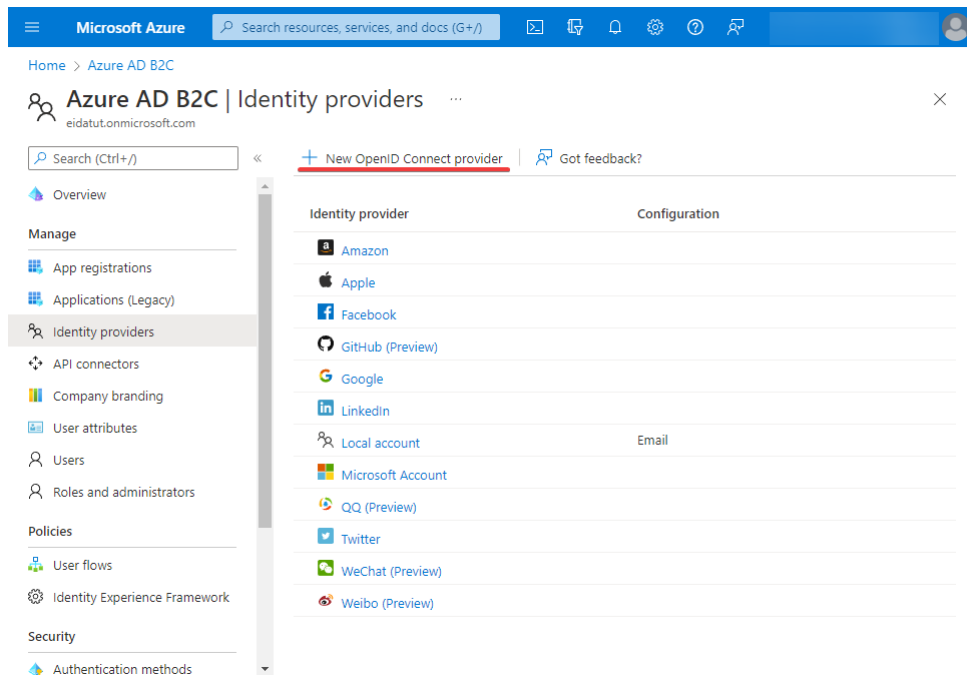


Figure 14. Adding OpenID Connect provider to Microsoft Azure

**Manual integration** If, for any reason, the use of libraries is not available or acceptable, developers can integrate the OpenID Connect protocol themselves. They do not have to integrate the whole protocol, just the code flow. For reference, we will use the source code from the .NET library [?] and the TARA documentation [?]. The steps listed will be as seen in figure ??.

**Login** This step is all about user agent session binding and authorize URL generation. This is a multipart section so each part will be split into its own paragraph.

For the authorize endpoint URL generation is best to look at chapter 4.1 of TARA documentation [?]. There are, however, some notes to keep in mind when integrating the eeID service:

- `https://auth.eeid.ee/oidc/.well-known/openid-configuration` contains all of the necessary information to realize which endpoints to use and what values are currently accepted.
- `https://test-auth.eeid.ee/oidc/.well-known/openid-configuration` contains the configuration for the eeID test environment.
- `response_type`: the only flow supported by TARA/eeID is code flow, so this value can only be code.

- `nonce`: although not required, it protects against authorization code injection attacks and should be used whenever possible [?]. See previous chapter for more details.
- `ui_locales`, `acr_values`: these fields are non-standard, and most if not all libraries will not support them out of the box. For most use cases, these values can be left empty.
- `code_challenge`, `code_verifier`: Proof Key for Code Exchange (PKCE) is not supported, `nonce` should be used in its place.
- `response_mode`: not supported, therefore only response mode of query can be used.
- `nonce`, `state`: both `nonce` and `state` have the exact behavior (see Discovered Weaknesses section for more information).

With this information, we can extrapolate a working authorize request (see listing ??).

---

```
GET https://test_auth.eeid.ee/oidc/authorize?

client_id=oidc_b0669946_896b4cdf_a478_a60afd5c18a5_20&
redirect_uri=https%3A%2F%2Fauth.eid.gedas.dev%2Fsignin_tara&
response_type=code&
scope=openid&
nonce=CWWLeSzn5tyu3XCSUTIz_BQJgnFxu7US&
state=J5BpaPNynnbhZCWmDlCZc5QWznVyIfebYGkZ3 ...
```

---

Listing 1. The eeID service authorize endpoint request

Before redirecting the user to the authorize endpoint, it is essential to bind the `nonce` and the `state` to the user agent session. We accomplish this by attaching an encrypted cookie to the response, which the user agent will save before it gets redirected.

**Authorize and Second log-in** When the user agent redirects the user to the eeID service authentication page, they can choose any authentication option. This process is not relevant to the thesis.

**Callback: verify state** After the user finishes authentication with the eeID service, they are redirected to the `redirect_url` defined in the request with particular query parameters (see listing ??). Note the inclusion of `nonce`, as this value should not be here [?] (see Discovered Weaknesses section), and we will ignore it for the rest of this section.

---

```
GET https://auth.eid.gedas.dev/signin tara?

code=OC 106 2hUkp91Z2acDYJF7PUFjDoTJKkHncVYl&
nonce=CWWLeSzn5tyu3XCSUTIz_BQJgnFxu7US&
state=J5BpaPNynnbhZCWmDICZc5QWznVyIfebYGkZ3...
```

---

#### Listing 2. The eeID service authorize redirect response

The WorkAuth's server's first step is to verify if the state received in the callback matches the one stored in the user agent session. If it does not, it was a possible CSRF attack, and the process should end here.

**Callback: acquire token** Once the authorization server that the state matches the user agent session, it is safe to exchange the received code for an identity token. See listing ?? for a request example.

---

```
POST /oidc/token HTTP/1.1
Host: tara.ria.ee
Content-Type: application/x-www-form-urlencoded
Authorization: Basic b2lkYyliMDY2OTk0Ni04OTZiLTRjZGYtYTQ3OC1hNjBhZmQ1YzE4YTUtMjA6aHR0cHM6Ly95b3V0dS5iZS9kUXc0dzlXZ1hjUQ==

grant_type=authorization_code&
code=OC 106 2hUkp91Z2acDYJF7PUFjDoTJKkHncVYl&
redirect_uri=https%3A%2F%2Fauth.eid.gedas.dev%2Fsignin tara
```

---

#### Listing 3. The eeID service token request

The inclusion of `redirect_uri` here may be confusing. It is required as per the OAuth2 spec (section 4.1.3) [?] and prevents open redirection attacks when using wildcards. The eeID service does not appear to allow them, so the inclusion of this value is redundant; however, if it were excluded, the protocol would no longer be OpenID Connect compliant.

**Callback: verify token** In the unlikely event the token endpoint request failed, it could mean a sophisticated replay attack could have taken place. If an access token was already issued for that code, it must be immediately revoked. Another way for the request to fail would be if the client or user agent took too long to be redirected. If the token endpoint returns a faulty result, the authentication process should stop.

If the token endpoint returns the identity token successfully, the user should validate its authenticity. If the nonce parameter was used in the first request, the server should verify that nonce in the user agent session matches the one in the `id_token`. The rest of the verification should be done as described in the TARA documentation [?].

I have no clue about the benefit of performing additional verification. The token was received within a secure backchannel, and the nonce matches, meaning the user created the request. If the server makes the user agent session in a way so that no one other than the server itself can tamper, why bother verifying the rest?

**Issue access token** After the server verifies the identity token, issue a new access token with the necessary information from the identity token. The most common solution would be to create a new cookie and attach it to the response.

## 5.7 Discovered Weaknesses

### 5.7.1 Incorrect implementation of at\_hash

In the TARA Technical Specification [?], the identity token has at\_hash value that is not according to the OIDC spec [?].

When looking at the id token response, it has a property at\_hash with the value of X0MVjwrmMQs/IBzfU2osvw==. This value is supposed to be base64url encoded. Instead, it is a regular base64 string. The Demo REST Client example provided by the same authors [?] correctly converts the base64 value into the base64url encoded value, which leads us to believe that there is a mistake in the documentation and or implementation. If it was following the specifications of the TARA documentation and not OIDC spec, it should have no reason to do so.

The eeID service follows the TARA documentation, and because of that, the at\_hash uses base64. An issue arises when the using OpenID Connect libraries (see listing ??).

---

```
IDX21348: Validating the 'at_hash' failed , see inner exception .
IDX21300: The hash claim: 'UtsKV8+hA/bB0EE/xR9cCQ==' in the
id_token did not validate with against: 'AT 95
VU6Y2LZjrNrVCdhIEaCxG6Gpzt0RsE Z', algorithm: 'RS256'.
```

---

Listing 4. Microsoft.IdentityModel.Protocols.OpenIdConnect fails to validate at\_hash

If we compute the hash manually, we see precisely why the verification failure happens (see listing ??). The implementation expects a different string than was provided. The same transcoding behavior is seen on the TARA Demo Client [?].

---

```
user@localhost:~$ access_token="AT 95
VU6Y2LZjrNrVCdhIEaCxG6Gpzt0RsE Z"
user@localhost:~$ echo n $access_token | openssl dgst -binary
sha256 | head -c 16 | base64
UtsKV8+hA/bB0EE/xR9cCQ==
user@localhost:~$ echo n $access_token | openssl dgst -binary
sha256 | head -c 16 | base64 | tr '+ ' '_' | tr -d '='
UtsKV8 hA_bB0EE_xR9cCQ
```

---

---

### Listing 5. Verifying at\_hash manually

The implication of this discovery means that all working clients who use eeID have incorrect OpenID connect implementation. This issue affects only those who use the correct OpenID Connect implementation libraries.

The reason for the incorrect implementation stems from backward compatibility [?].

#### 5.7.2 Confusing state and nonce behavior

Nowhere in the OpenID Connect specification mentions that state should be transferred over to the id\_token. TARA confused the purposes of state and nonce properties, extended the behavior to cover each other, and, by extension, made one of the properties obsolete.

The state property is part of the underlying OAuth2.0 specification, where it is an "opaque value used by the client to maintain state between the request and callback" [?]. The primary security feature is to prevent CSRF attacks [?, ?].

The nonce property is part of the OpenID Connect specification, and its primary purpose is to prevent replay attacks when using implicit or hybrid flows [?]. Later, researchers discovered that it could also protect against authorization code injection attacks with the code flow. The disadvantage of using nonce as a state parameter is that it directly influences the size of the id\_token, which should be kept as small as possible. The reason for keeping this token as small as possible is so that developers could later send them in request headers to their resource servers [?]. The only issue is that these tokens expire after 40 seconds and cannot be refreshed [?], making this approach impractical.

If we look at the data flow diagram (see figure ??), we see that both state and nonce have each other's properties. After the user agent returns to the callback URL, both nonce and state are returned when only state is required. After the company's authentication server establishes a backchannel and redeems the code for an id\_token, this token again contains both nonce and state when only nonce is required.

In the security analysis performed last year, a researcher suggested removing the nonce parameter from the protocol [?]. We disagree with this approach and would suggest removing the state parameter from the id\_token response. Removing the nonce parameter and having no support for PKCE would break OIDC compliant libraries' ability to mitigate authorization code injection attacks.

#### 5.7.3 Wrong claims in the OpenID Connect discovery endpoint

The discovery endpoint provides all information about possible requests and responses. The data listed there does not match the documentation. For example, in the discovery endpoint for eeID (<https://auth.eeid.ee/oidc/.well-known/openid-configuration>),

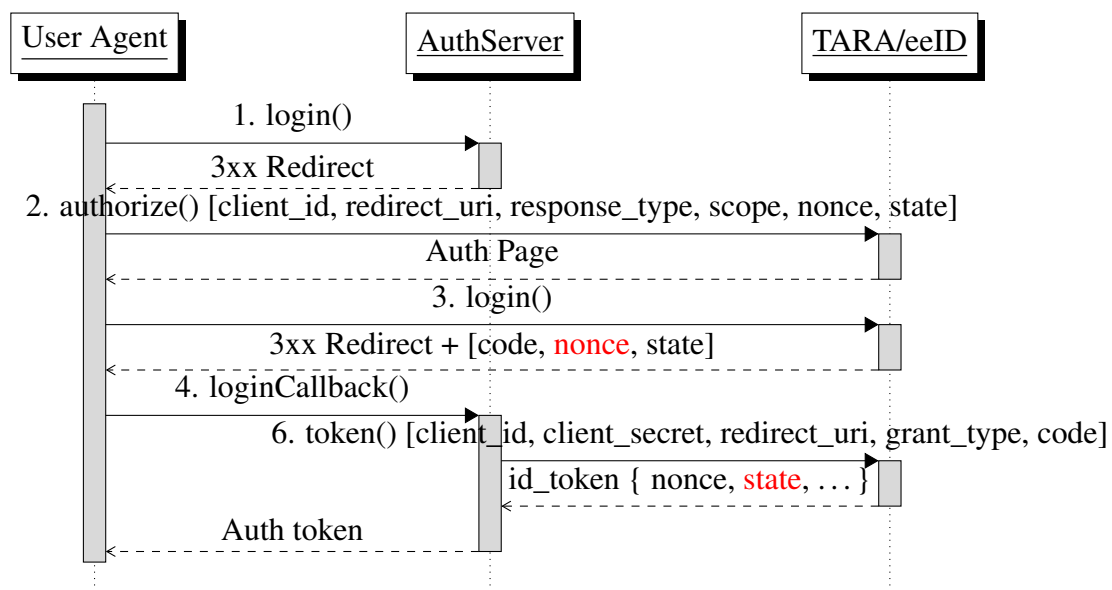


Figure 15. The incorrect OIDC code flow used in TARA/eeID

claims like gender are present, even though this claim can never appear. On the flip side, the claim `profile_attributes`, as described in the documentation, is missing from the discovery document.

should I include token and .well-known response example here?

## 6 Case Study: Dokobit

### 6.1 About

Dokobit [?], trademark and subsidiary of Estina [?], offers two products: Document Signing Portal and API solutions.

The first product, the Document Signing Portal, was released in 2014 [?]. The primary purpose of this solution is to allow users to upload documents and digitally sign them online.

Estina has acquired the DigiDoc portal (<https://digidoc.ee>) from SK ID Solutions in 2016 [?] which had the exact purpose. This portal should not be confused with the similarly named DigiDoc4 client Estonians commonly use to sign documents [?].

The second product, API solutions, targets businesses in a variety of scopes: signature collection, signing, identification, sealing, and TSP monitoring [?]. In the thesis, we will only consider the Identification service.

Dokobit's Identification service allows Lithuanian, Latvian, Estonian, Finnish, Norwegian, Icelandic, Polish, Belgian, Portuguese, Spanish, and Italian [?] users to authenticate themselves with their countries' scheme.

The company has received ISO/IEC 27001:2013 certification [?] and in 2020 was included in the EU Trusted Service List [?, ?], thus it becoming a Qualified Trust Service Provider.

In 2021, the Norwegian electronic identity solutions provider company Signicat AS acquired UAB Dokobit.

**Dokobit Identification Service** Identification service supports two distinct data flows: Gateway and API. The core difference between them is that the Gateway uses a prebuilt UI on their server. In contrast, the API requires the companies to develop their UI and have their server communicate with Dokobit servers instead. This difference only affects the user experience.

The main advantage of Identity Gateway over Identity API is the added brand trust. A study finds that users "associate higher security feelings with a higher level of brand trust" [?]. If an organization has not matured yet as a brand (such as a recent startup), it will make more sense for them to choose Identity Gateway over API. On the contrary, if they are a large, highly trusted organization, such as a bank, it would make more sense to use Identity API and have all user interactions happen on the same domain.

For this thesis, we will only analyze the Identity Gateway.

**Embed or Redirect** Identity Gateway comes in two primary user flows: embedded and redirect-based.

In embedded flow, users could stay on the website, authenticate in a pop-up window, and update the website view accordingly after finishing the authentication process.



Embedded flow has the advantage of not requiring the users to leave the website, which is helpful to preserve user data in complex forms.

In redirect-based flow, users are sent to an external website, perform authentication, and redirected back to the company website, in a flow similar to OAuth2.0.

Ultimately, experts consider the embedded flow to be the weaker of the two methods [?] for two main reasons:

1. Cross-origin requests are inherently more dangerous, allowing for MitM and CSRF attacks;
2. The client application, even when embedded, receives full client credentials, which adds another point of compromise in the form of XSS;

When using federated log-in for Native Apps, "best current practice requires that native apps **MUST NOT** use embedded user-agents to perform authorization requests" [?]. This practice means that companies who have a mobile app or would consider having one in the future mustn't use the embedded flow.

## 6.2 Data Flow

This section will analyze Dokobit Identity Gateway, redirect-based user flow. The general overview of which can be seen in figure ??. We can group the authentication process into three parts: establishing a session with Dokobit, user authentication with an eID provider, and user information retrieval.

**Establishing a session with Dokobit** When a user requests to authenticate, the company's back-end systems' first step is to establish a session with Dokobit Identity Gateway. To do this, a POST request must be made to the /api/authenticate/create endpoint. This response will contain the session identifier and the redirect URL. Users will have to go there to interface with their eID providers.

Sample HTTP request data can be seen in listing ??.

---

```
Request :
POST https://id sandbox.dokobit.com/api/authentication/create?
    access_token=YOUR_ACCESS_TOKEN
{
    'return_url': 'https://id sandbox.dokobit.com/example/success.
    php'
}

Response :
{
    "status": "ok",
```

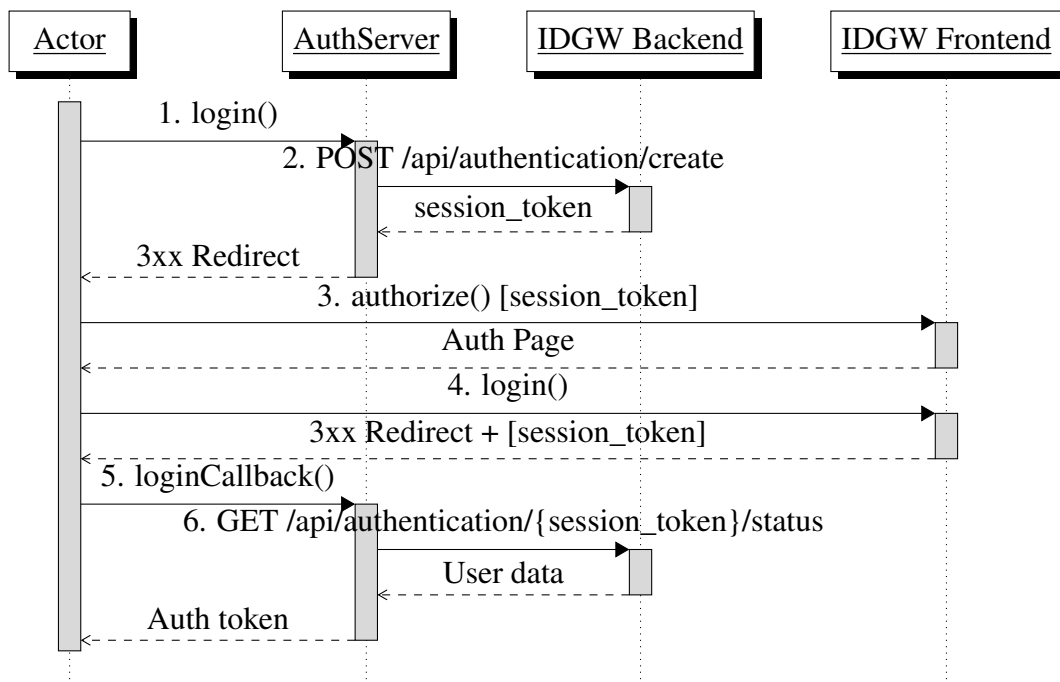


Figure 16. Dokobit Identity Gateway - Redirect-based user flow [?]

```

"session_token": "02
f922c9917231ea8acbbbcf63796924af548c801d75772f2b1701b413462c61
",
"url": "https://id_sandbox.dokobit.com/auth/02
f922c9917231ea8acbbbcf63796924af548c801d75772f2b1701b413462c61
",
"expires_in": 3600
}

```

Listing 6. Handling Dokobit session creation

**User authentication with an eID provider** After the back-end successfully creates a session, they must redirect the user to the received endpoint. An easy way to accomplish that is to respond to the initial authentication request with HTTP status 302 - Found.

Most of the heavy lifting with authentication is delegated to this step and handled by Dokobit. The company's back-end systems should wait for the user to return after authenticating.

**User information retrieval** After the user returns after successful authentication, the back-end servers should make a GET request to /api/authentication/session\_token/status endpoint. The company can securely receive the user information via a backchannel.

Sample HTTP request data can be seen in listing ??.

---

```
Request:
GET https://id sandbox.dokobit.com/api/authentication/SESSION_TOKEN
/status?access_token=YOUR_ACCESS_TOKEN

Response:
{
  "status": "ok",
  "certificate": { ... },
  "code": "30303039914",
  "country_code": "lt",
  "name": "DEMO",
  "surname": "SMART ID",
  "authentication_method": "smartid",
  "date_authenticated": "2019 05 06T12:15:34+03:00"
}
```

---

Listing 7. Handling Dokobit session creation

## 6.3 Trust Anchor

Dokobit assumes the role of being the trust anchor. This means that it also acts as a single point of failure. Should Dokobit become compromised, all applications using Dokobit will become susceptible to impersonation. On the flip side, there is only one system company developers need to implement, meaning that it would be harder for adversaries to break into the system via external means.

Companies should consider the risks when using a provider capable of dictating who the person is, as no integrity checks are supported in the protocol.

From a technical aspect, these risks aren't much different from those applicable to the eeID service.

## 6.4 Pricing

Dokobit is a commercial product, and therefore it has associated usage costs. In 2022, these costs are as seen in the table ??.

Table 2. Dokobit Identity Gateway pricing 2022

Plan	Number of transactions	Monthly fee	Price per extra transaction
1	700	50 €	0,071 €
2	1 600	100 €	0,063 €
3	5 000	250 €	0,050 €
4	12 000	500 €	0,042 €

Each pricing tier includes a specific number of transactions and adds a cost for each transaction exceeding it. For example, if the total amount of transactions is 200, the price will still be 50 €, as the number of transactions has not reached 700.

Assuming the company's users authenticate around 25 times per month, the monthly user price will be in the ballpark of 1-2 €.

## 6.5 Security Requirements

Of the three case studies analyzed in the thesis, Dokobit Identity Gateway was the only one that did not provide any validation requirements to the relying party, only a single integration example [?]. This flow, however, is very similar to the one used by the eeID service and will be analyzed with the same best practices document [?].

**Communication channel** The Dokobit identity gateway uses a secure communication channel, encrypted end-to-end using HTTPS. When the system is integrated correctly, malicious clients (or user agents) have no possible way of influencing any of the authentication parameters without triggering alarms when validating the request.

### 6.5.1 Protocol's built-in security features

#### Replay attacks

- Identity Gateway will reject the second GET request with the session id.

The developer does not have to implement state management to verify that a given session token was used only once. Mitigation measures are sufficient.

#### Insufficient Redirect URI Validation

- The adversary does not have any agency over redirect URI after authentication, as the authorize endpoint does not have any query parameters.
- The redirect\_url parameter when sending the initial request outlined in step 3 of the sequence diagram cannot be longer than 255 ASCII characters.
- The redirect URL is never validated on the Dokobit servers, as it was never registered as a client. Each redirect URL is generated for one-time use.

Adversaries cannot influence the redirect URI. If companies use a static redirect URL, mitigation measures are sufficient.

Name	Status	Domain
ExternalLogin?returnUrl=%2F	302	auth.eid.gedas.dev
0ccb289b93211e315e9a44d7bed...	200	id-sandbox.dokobit.com
login.css	200	id-sandbox.dokobit.com
identity.css	200	id-sandbox.dokobit.com
applet.css	200	id-sandbox.dokobit.com
app.js	200	id-sandbox.dokobit.com
translations.js?v=1648728361754	200	id-sandbox.dokobit.com
favicon.ico	200	id-sandbox.dokobit.com
ic_smartid_24px.svg	200	id-sandbox.dokobit.com
ic_eparaksts_mobile_24px.svg	200	id-sandbox.dokobit.com
ic_audkenni_app_24px.svg	200	id-sandbox.dokobit.com
lt.svg	200	id-sandbox.dokobit.com
lv.svg	200	id-sandbox.dokobit.com
ee.svg	200	id-sandbox.dokobit.com
Roboto-Medium.ttf	200	id-sandbox.dokobit.com
Roboto-Regular.ttf	200	id-sandbox.dokobit.com
smartid?_locale=en	200	id-sandbox.dokobit.com
status?_locale=en	200	id-sandbox.dokobit.com
status?_locale=en	200	id-sandbox.dokobit.com
status?_locale=en	200	id-sandbox.dokobit.com
0ccb289b93211e315e9a44d7bed...	302	id-sandbox.dokobit.com
signin-dokobit?session_token=0c...	302	auth.eid.gedas.dev

Figure 17. The Dokobit Identity Gateway does not use resources outside of their domain in the authentication flow

### Credential Leakage via Referrer Headers

- Dokobit Identity Gateway does not include third-party resources (javascript, image, or other); therefore, it cannot leak the session token.
- The company is required not to have any third-party resources on the authentication and redirect pages.

Dokobit does not leak credentials via Referrer Headers. The developers should not embed third-party resources in the critical authentication pages (all resources used in their service are from their domain, see figure ??). Mitigation measures are sufficient.

### Credential Leakage via Browser History

- Session ids are stored in browser history (see figure ??); however, they are single-use only and are immune to replay attacks.

The only thing adversaries could extract from the browser history is a used-up session-id, which does not provide much value with sufficient CSRF measures. Mitigation measures are adequate.

**Session Token Injection and Cross-Site Request Forgery** Adversaries would perform the injection and CSRF attacks identically.

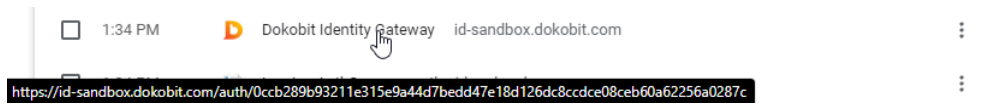


Figure 18. Session id is leaked inside the browser history when using Dokobit authentication

- The protocol does not protect against session token injection attacks.
- The protocol also adds more attack surfaces by issuing the session token before the user signs in, allowing the adversary to obtain or inject it before and after the user performs authentication.
- It is possible to mitigate adversaries stealing a victim's session.
- **It is impossible to mitigate adversaries injecting their session** (phishing).

Identity Gateway protocol does not have security measures built-in against session token injection or CSRF attacks. It is possible to prevent adversaries from being able to redeem the victim's session. However, there is **no possible way to prevent adversaries from injecting their Dokobit session**.

From the attacker's point of view, they can exploit this vulnerability by performing these steps [?]:

1. Establish session with identity server and Dokobit;
2. Trick the user into opening the link issued by Dokobit;
3. After the user authenticates, but before gets redirected, reload the page;
4. If an adversary managed to do it in time, they receive that person's access;

The only way to mitigate this attack for the user will be to make the authentication server invalidate the authenticated session if a request presents the same session\_id. Thankfully, it is easy to detect if the server has consumed the session token already, as Dokobit would reject additional HTTP requests, as described in replay attack mitigation.

Mitigation measures are **weak**, as the Dokobit's internal log-in page is susceptible to CSRF attacks, and developers should be ready to implement authentication session revocation.

```
▼ Response Headers
access-control-allow-headers: X-CSRF-Token
access-control-allow-origin: https://auth.eid.gedas.dev
cache-control: max-age=0, must-revalidate, private
content-encoding: gzip
content-type: text/html; charset=UTF-8
date: Sat, 02 Apr 2022 10:53:38 GMT
expires: Sat, 02 Apr 2022 10:53:38 GMT
server: nginx
strict-transport-security: max-age=63072000; includeSubDomains
x-frame-options: SAMEORIGIN
```

Figure 19. Response headers browsers receive when opening the eeID service’s login page

### Clickjacking

- Dokobit auth page does not use Content-Security-Policy. However, it does use header X-Frame-Option: SAMEORIGIN (see figure ??).
- The sameorigin feature is used in conjunction with CORS to support embedded flow, however it would be better if Frame Options were outright disabled.
- Relying party should integrate this or a similar countermeasure.

Mitigation measures are sufficient on almost all browsers released in the last ten years [?].

### 6.5.2 Validation requirements for the relying party

In the previous section, we saw the security features of the protocol. The relying party should implement validation and reject requests for some of the features if those validations do not match. This section will contain all of the features the RP needs to address.

### Misconfiguration Attacks

**Incorrect URL specified** The relying party should make sure to use the correct Dokobit Identity Gateway endpoints. For test environment the URL is `https://id-sandbox.dokobit.com`, and for production - `https://id.dokobit.com`. A connection with a malicious party could be established when using the incorrect URL.

Developers should make sure these values are not easily editable (such as placed in environment variables) by anyone. Best they should be hardcoded in the application.

**Weak TLS configuration** TLS is the primary defense mechanism against MitM attacks when connecting to the Dokobit servers. WorkAuth's IT Ops team should ensure that the server does not trust any malicious CA.

**Session Token Injection and Cross-Site Request Forgery** Like in the eeID service, the primary validation user has to implement is protection against CSRF and code injection. Developers can accomplish this by binding the dokobit session\_id to the user agent session.

**Attack description** For reference to how the mitigation works, we will use the figure ???. We will provide two examples for the countermeasure: without and with session binding.

Our attack exists right before the final redirect (right before step 5 in the figure ??). Unlike in the case with eeID, this protocol has multiple places where an attacker can extract the session\_id as it is generated before the user is redirected, and not after they complete the authentication process. Regardless, an attacker cares about tokens that have been authenticated, but not consumed, which puts us right before step 5.

**Attack #1 - No binding** When a subsequent request is made to the authorization endpoint, if the session\_id was authenticated, Dokobit will automatically redirect the user to WorkAuth's identity server and finish the process. Without user agent binding, it is trivial to establish a session.

**Attack #2 - With user agent binding** Before WorkAuth's server redirects the user agent to Dokobit's authorization endpoint, it first creates a cookie containing the session\_id in an encrypted form. This value will be used later.

We run the first attack per usual, interrupting the flow in the same place. When the attacker tries to authenticate, the server will successfully decrypt the cookie value. However, the server will fail to match the session\_id and detect the attack.

Because everything is bound to the same session\_id, this one cookie is sufficient to mitigate against both CSRF and code injection attacks.

## 6.6 Integration

Unlike the eeID service, no libraries exist for Dokobit Identity Gateway, and users must manually integrate the flow. Fortunately, the integration is very straightforward.

The code snippets ?? and ?? are akin to data flow in figure ??. Numbers in comments directly correspond to the steps in the flow.

The first code block (listing ??) has three main parts:



1. establish a session with Dokobit as described in listing ??;
2. append an **encrypted** session cookie which would bind the browser to that specific Dokobit session (this cookie stores the encrypted session\_token);
3. redirect the user to the authorization endpoint;

---

```
// [1] Login
protected override async Task HandleChallengeAsync(
    AuthenticationProperties properties)
{
    if (string.IsNullOrEmpty(properties.RedirectUri))
        properties.RedirectUri = OriginalPathBase + OriginalPath +
            Request.QueryString;

    // [2] Establish session with Dokobit
    var body = JsonSerializer.Serialize(new { return_url =
        BuildRedirectUri(Options.CallbackPath) });
    var response = await ExecuteRequestAsync(HttpMethod.Post, "/api
        /authentication/create", body);
    response.EnsureSuccessStatusCode();

    using var sessionResponse = await JsonDocument.ParseAsync(await
        response.Content.ReadAsStreamAsync(Context.RequestAborted))
        ;
    var root = sessionResponse.RootElement;

    // Save session token to encrypted cookie properties
    properties.Items["session_token"] = root.GetString("
        session_token");

    Response.Cookies.Append(Options.StateCookie.Name!, Options.
        StateDataFormat.Protect(properties), Options.StateCookie.
        Build(Context, Clock.UtcNow));
    var redirectContext = new RedirectContext<DokobitOptions>(
        Context, Scheme, Options, properties, root.GetString("url")
        !);

    // [3] Redirect the user agent
    await Events.RedirectToAuthorizationEndpoint(redirectContext);
}
```

---

Listing 8. Handling Dokobit session creation

After the user authenticates and returns, the next code block gets executed (listing ??). This code performs request validation in addition to retrieving the user data:

1. try to decrypt the cookie data - if decryption fails, it was likely tampered with, and the application cannot proceed with the authentication;

2. try to obtain the session token from query parameters - it should always exist if the request was not tampered with;
3. verify that the session\_token received from cookie and query parameters match - this prevents attackers from stealing session tokens;
4. redeem the session\_token and receive user data as described in listing ??;
5. clean up the used cookies;
6. issue an access token or a new session for use in the company;

---

```
// [5] Callback
protected override async Task<HandleRequestResult>
    HandleRemoteAuthenticateAsync()
{
    // Decrypt the user agent session. If we fail here, it means
    // that cookie was likely tampered with
    var properties = Options.StateDataFormat.Unprotect(Request.
        Cookies[Options.StateCookie.Name!]);
    if (properties == null)
        return HandleRequestResult.Fail("Invalid state");

    // Verify if session token received matches the one we
    // initially saved
    var sessionToken = Request.Query["session_token"];
    if (StringValues.IsNullOrEmpty(sessionToken))
        return HandleRequestResult.Fail("Missing session_token",
            properties);

    if (properties.Items["session_token"] != sessionToken)
        return HandleRequestResult.Fail("Unexpected session_token
            received", properties);

    // [6] Validation successful, obtain user identity
    var response = await ExecuteRequestAsync(HttpMethod.Get, $"/api
        /authentication/{sessionToken}/status");
    response.EnsureSuccessStatusCode();

    // Cleanup
    Response.Cookies.Delete(Options.StateCookie.Name!);

    // Establish an authenticated session
    await using var stream = await response.Content.
        ReadAsStreamAsync(Context.RequestAborted);
    using var user = await JsonDocument.ParseAsync(stream);
```



## Disclosure

- Mar 02, 2022: Informed Dokobit CIRT about the exploit.
- Mar 13, 2022: Dokobit CIRT acknowledged the issue, but deemed not worth fixing.

### 6.7.2 CSRF on the authorize endpoint

When performing a session token injection attack, relying parties can successfully protect themselves by binding the Dokobit's session token to the user agent session. Unfortunately, this does not prevent attackers from injecting their session tokens into victims' computers.

From the attacker's point of view, they can exploit this exploit can be achieved by performing these steps [?]:

1. Establish session with identity server and Dokobit;
2. Trick the user into opening the link issued by Dokobit (phishing works);
3. After the user authenticates, but before gets redirected, reload the page;
4. If an attacker manages to achieve it in time, they receive that person's access;

A video exists showing a live demonstration of how this attack appears for both attacker and victim [?].

**The underlying issue** At its core, this exploit is possible because of three issues:

1. The attacker knows the response session token. Because all requests are correlated with the same session token, the attacker already knows the redirect URL.
2. Nothing is binding the authentication and redirect processes.
3. Requests automatically redirect the user to the final endpoint after authentication if the session token was not yet consumed.

The third point is fascinating, as it glues issues 1 and 2 together. We can see how it connects these issues by looking at an authentication example (see figure ??).

We need to look at the waterfall diagram of the requests Identity Gateway makes to see why this attack works. The first request (smartid) starts a background task to authenticate with Smart-ID. Second and third requests (status) check on that background job and complete the request after it does. Notably, the checks are done with a delay, meaning

Name	Status	Domain	Type	Size	Time	Waterfall
<input type="checkbox"/> smartid?_locale=en	200	id-sandbox.dokobit.com	xhr	324 B	266 ms	
<input type="checkbox"/> status?_locale=en	200	id-sandbox.dokobit.com	xhr	360 B	1.36 s	
<input type="checkbox"/> status?_locale=en	200	id-sandbox.dokobit.com	xhr	355 B	296 ms	
4cab404b148f78d50e679ef549d...	302	id-sandbox.dokobit.com	docu...	1.1 kB	163 ms	
signin-dokobit?session_token=4...	302	auth.eid.gedas.dev	docu...	1.2 kB	243 ms	

Figure 21. Waterfall of Dokobit Smart-ID authentication

there could be some amount of time (around one second) when the authentication process is finished, but the browser is not aware of it.

When the page is refreshed, status checks are done immediately, so if an attacker managed to refresh the page in the short 1-second window, they would steal the session. Ironically, if the WorkAuth developers were diligent enough to prevent CSRF attacks, in the server's eyes, the victim would be seen as an attacker, and their request would be rejected, giving even more time for attackers to exchange the token.

There may be some band-aid solutions for this exploit, but we see only one true fix: bind the Smart-ID (and other) authentication schemes to the current request or make it synchronous. Regular CSRF countermeasures are sufficient (generating a token on the page and using it to the end).

## Disclosure

- Mar 21, 2022: Informed Dokobit CIRT about the exploit.
- Mar 27, 2022: Dokobit CIRT acknowledged the issue and started to work on a fix.

## 7 Case Study: Web eID

### 7.1 About

Released in the Summer of 2021 [?] and having undergone significant changes in January of 2022, this eID framework allows users to authenticate and sign documents using their country's smart cards.

Functionally this framework is split into three parts: software the user needs to install on their computer, a javascript library that acts as a data transfer intermediary, and the certificate validation library for the back-end.

The software users need to install is similar to the one various countries' governments issue. The significant difference is that this software supports more than one countries' eID solutions. Supported countries include Estonia, Latvia, Lithuania, and Finland [?].

This service is built by the Estonian Information System Authority, responsible for TARA, Estonia's public sector gateway. No publicly available audit certification is available at the time of writing .

is this  
true?

### 7.2 Data Flow

Figure ?? displays the high-level overview of the complete flow of data within the Web eID framework. A detailed explanation of the steps can be found on the technical specification page [?]. Companies implementing the framework should only consider the browser and the server application (steps 1-3 and 13-17).

Steps 1-3 ... Steps 13-17

### 7.3 Trust Anchor

Unlike Dokobit and eeID, Web eID does not provide any guarantees about the trust-worthiness of a certificate. It is, however, not out of malice and reminds developers by sending the certificate in a field called "unverifiedCertificate" [?].

The relying party must verify the certificate and challenge themselves by checking the origin, certificate expiry, trust chain, OCSP response, and the challenge. This validation structure makes the trust anchor technological and highly dependent on the implementation correctness by the developers.

### 7.4 Pricing

The Web eID authentication service is free of charge, as the only external validation, OCSP [?] requests, are free to use. When creating digital signatures, the timestamping service may require payment [?], however the focus is only on the authentication.

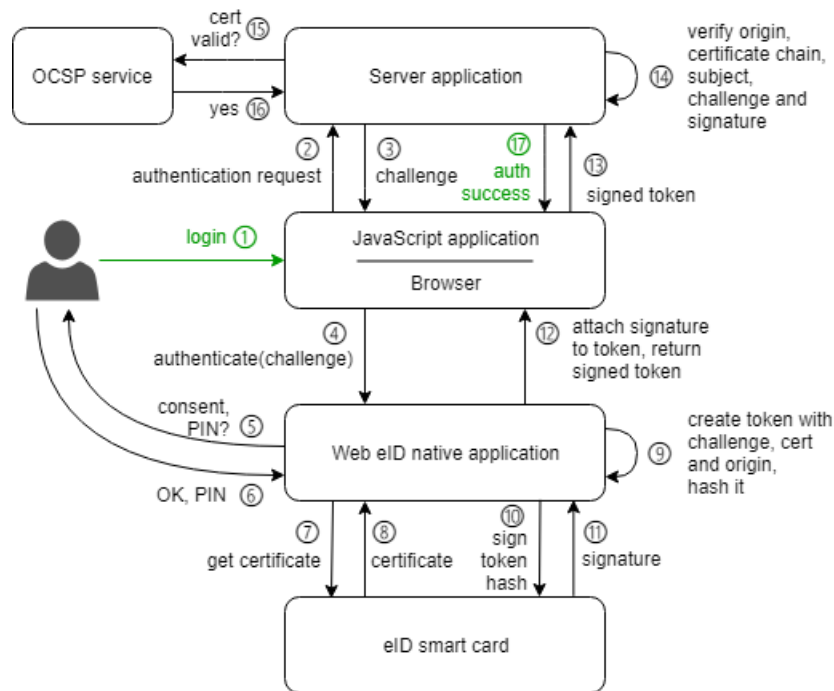


Figure 22. Web eID Authentication flow [?]

## 7.5 Security Requirements

Web eID is almost entirely an offline protocol; therefore, the IETF guidelines document [?] used for eeID, and Dokobit is not applicable here. Instead, we will solely rely on the implementation details provided by RIA [?].

The second version of the protocol (having accounted for the feedback of Arnis Paršovs [?]) will be analyzed.

**Communication channel** The Web eID framework is the only one covered by this thesis that uses an insecure communication channel. Developers must take caution and verify received data when implementing the framework. This channel is highly susceptible to man-in-the-middle, forgery, or other attacks done by the user.

Unlike in the cases of Dokobit and eeID, the risk of impersonation is not transferred to the eID service provider. However, a higher degree of certainty is provided, as only the client can be malicious. The underlying trust anchor, cryptography, is unlikely to be nefarious.

## 7.6 Integration

For each protocol implementation step, developers will have to fulfill certain guarantees before the system goes into production.

### 7.6.1 Preparation

Building the challenge nonce. The goal of these steps is to create the challenge the user will have to sign with their private key. There are a couple of guarantees the application must provide:

1. Generated challenge nonce must be between 32 and 96 bytes (inclusive) in length [?];
2. "It must be guaranteed that the authentication token is received from the same browser to which the corresponding challenge nonce was issued" [?]. The framework creators suggest attaching it to the user session.
3. "Cache must be used for protection against replay attacks by guaranteeing that each authentication token can be used exactly once" [?].
4. "Cookie-based authentication must be protected against cross-site request forgery (CSRF) attacks and extra measures must be taken to secure the cookies by serving them only over HTTPS and setting the HttpOnly, Secure and SameSite attributes" [?].

In the implementation example, these measures were addressed by:

1. a 64 byte cryptographically secure randomly generated nonce is created (see listing ??);
2. challenge nonce is set in the user's session, which adversaries cannot tamper;
3. the generated nonce is stored into local memory cache for later use; nonce expires after 5 minutes;
4. an input field is rendered on the page with a unique CSRF validation token, which prevents cross-site request forgery attacks (see listing ??);

---

```
private TimeSpan ChallengeLifetime { get; } = TimeSpan.FromMinutes(5)
;

private readonly IMemoryCache _cache; // Injected

[HttpGet("challenge")]
```



```

public IActionResult GetChallenge()
{
    var nonce = RandomNumberGenerator.GetBytes(64);

    _cache.Set(Convert.ToBase64String(nonce), true, ChallengeLifetime);
    HttpContext.Session.Set("eid.challenge", nonce);

    return Ok(new { nonce });
}

```

---

Listing 10. Web eID Challenge Endpoint

---

```

@Inject Microsoft.AspNetCore.Antiforgery.IAntiforgery _csrf
@{ var csrfToken = _csrf.GetAndStoreTokens(HttpContext); }

<!-- Button used to sign in -->
<a role="button" class="btn btn secondary" id="webeid_auth_button">
    Web eID</a>

<input id="csrfToken" type="hidden" value="@csrfToken.RequestToken"/>

<script>
    ...

    const authTokenResponse = await fetch("/signin_id/login", {
        method: "POST",
        headers: {
            "Content Type": "application/json",
            "RequestVerificationToken": document.getElementById("
                csrfToken").value
        },
        body: JSON.stringify(...)
    });

    ...
</script>

```

---

Listing 11. Web eID UI excerpt

---

### 7.6.2 Validation

After the user signs the nonce challenge and sends their certificate, the server must verify its authenticity. The application must perform all of the following before allowing the user to sign in:

1. verify the CSRF token from earlier steps [?];

2. verify the challenge nonce came from the original user and has not expired, was not consumed;
3. verify the certificate validity and check if nonce was signed by the associated private key (see below);
4. issue an authentication token with the fields from the certificate's subject;

In the implementation example, these measures were addressed by:

1. the back end endpoint for log-in is decorated with `ValidateAntiForgeryToken` Attribute. This attribute instructs the ASP.NET API to ignore requests not containing a CSRF token [?]. A JavaScript application can only access the protected endpoints by providing `RequestVerificationToken` header (see listing ??);
2. the application tries to extract the challenge nonce from the browsing session. The process would succeed if the session cookie were not modified. After the extraction, the application checks the nonce cache to verify if the challenge is still active. Cache hit means the nonce has not expired, and no previous authentication attempt was performed. Remove the challenge nonce from all stores.
3. The API calls a standalone validation service to verify the nonce and certificate (see certificate and nonce verification paragraph below).
4. Application populates the ASP.NET identity management system with the fields from the certificate: serial number, given name, surname, country. An identity session cookie is sent to the client.

---

```
[HttpPost("login")]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Login([FromBody]
    WebIdAuthTokenResponse token)
{
    // Obtain the challenge from session
    if (!HttpContext.Session.TryGetValue(ChallengeNonceKey, out var
        nonce) && nonce == null)
        return Unauthorized();

    // Check if token was not used before or expired
    var challenge = Convert.ToBase64String(nonce);
    if (!_cache.TryGetValue(challenge, out _))
        return Unauthorized();

    _cache.Remove(challenge);
    HttpContext.Session.Remove(ChallengeNonceKey);
}
```

```

// Validate the certificate and signed challenge
var validationResult = await _webEidValidationService.GetResult(
    new WebEidValidationRequest(token, nonce));
if (!validationResult.Success)
    return Forbid();

// Certificate is valid. Sign in the user

await HttpContext.SignInAsync(BuildUser(new X509Certificate2(
    Convert.FromBase64String(token.UnverifiedCertificate)).Subject
));

return Ok();
}

```

---

Listing 12. Web eID Login Endpoint

### 7.6.3 Certificate and nonce verification

This step is the most complicated in the entire validation process. To prevent any issues with incorrect implementation, the framework maintainers recommend using their library for validation [?]. Libraries can come with security vulnerabilities, and developers are reluctant to update their used version; however, it is still more favorable to creating vulnerabilities from misconfiguration [?].

The eu.webeid.security Java package performs most of the certificate validation: expiry, purpose, policy, OCSP [?]. Developers will only have to configure the CA and host validation. Configuration is handled by providing a set of trusted CA certificates for trust chain verification and the hostname for challenge nonces (see listing ??).

---

```

public class AuthTokenValidatorService {

    @Bean
    public AuthTokenValidator validator() {
        try {
            return new AuthTokenValidatorBuilder()
                .withSiteOrigin(URI.create(System.getenv("ORIGIN_URL")))
                .withTrustedCertificateAuthorities(
                    loadTrustedCACertificatesFromCerFiles())
                .build();
        } catch (JceException e) {
            throw new RuntimeException("Error building the Web eID auth
                token validator.", e);
        }
    }

    private X509Certificate[] loadTrustedCACertificatesFromCerFiles() {
        List<X509Certificate> caCertificates = new ArrayList<>();
    }
}

```

```

try {
    CertificateFactory certFactory = CertificateFactory.getInstance(
        "X.509");

    File[] files = new File("/certs").listFiles((f, n) > n.
        endsWith(".cer"));
    if (files != null) {
        for (File file : files) {
            try (InputStream stream = new FileInputStream(file)) {
                X509Certificate caCertificate = (X509Certificate)
                    certFactory.generateCertificate(stream);
                caCertificates.add(caCertificate);
            }
        }
    }
} catch (CertificateException | IOException e) {
    throw new RuntimeException("Error initializing trusted CA
        certificates.", e);
}

return caCertificates.toArray(new X509Certificate[0]);
}
}

```

---

Listing 13. Web eID Login Endpoint

The token validation service described in listing ?? requires WorkAuth maintainers to set the origin URL in the form of an environment variable and to populate the folder /certs with trusted CA certificates.

Origin URL can be obtained by checking the window.origin JavaScript variable in the page containing the log-in button.

For the CA certificate set, the company can get an up-to-date list of trusted certificates at the EU Trust Services Dashboard [?]. The issue with this list is that it contains all trust certificates for various scopes. In our case, we should limit the search to the extent of QCert for ESig. In the case of Estonia and Lithuania, only three entities are certified to issue certificates for QSCD (see figure ??). It is in stark contrast to Spain's 31 [?]. It is possible to further narrow down to only certificate generation services for qualified certificates (CA/QC).




In the case of Estonia's single TSP, we can see that only 3 CA are currently operational (see figure ??). Unfortunately, there is no standardized way of narrowing down which certificates could be used for authentication.

An alternative way to obtain certificates would be to go to the government authority of each country responsible for the distribution of certificates. This action requires prior knowledge of who is responsible for issuing certificates and their purposes.

In Lithuania's case, it is the Ministry of the Interior [?] who issues two certificates

**Trust service providers** results(5) Share New Search

Currently active trust service providers

-  **SK ID Solutions AS** QCert for ESig QCert for ESeal QTimestamp Cert for ESig Cert for ESeal
-  **Identity Documents Personalisation Centre under the Ministry of the Interior** QCert for ESig
-  **State Enterprise Centre of Registers** QCert for ESig QCert for ESeal QTimestamp

Current search

Qualified certificate for electronic signature




-  Estonia
-  Lithuania

Figure 23. List of EU Trust service providers of Estonia and Lithuania capable of creating qualified certificates for e-signatures

 **SK ID Solutions AS**

**Trust services**

QCert for ESig Qualified certificate for electronic signature

EID-SK 2007 <span>CA/QC</span> <span>Withdrawn</span>	EID-SK 2007 OCSP RESPONDER <span>OCSP/QC</span> <span>Withdrawn</span>
EID-SK 2007 OCSP RESPONDER 2010 <span>OCSP/QC</span> <span>Withdrawn</span>	EID-SK 2011 qualified certificates for electronic signatures <span>CA/QC</span> <span>Withdrawn</span>
EID-SK 2016 qualified certificates for electronic signatures <span>CA/QC</span> <span>Granted</span>	ESTEID qualified certificates for electronic signatures (ESTEID-SK 2011) <span>CA/QC</span> <span>Withdrawn</span>
ESTEID qualified certificates for electronic signatures (ESTEID-SK 2015) <span>CA/QC</span> <span>Granted</span>	ESTEID qualified certificates for electronic signatures (ESTEID2018) <span>CA/QC</span> <span>Granted</span>
ESTEID-SK <span>CA/QC</span> <span>Withdrawn</span>	ESTEID-SK 2007 <span>CA/QC</span> <span>Withdrawn</span>
ESTEID-SK 2007 OCSP RESPONDER <span>OCSP/QC</span> <span>Withdrawn</span>	ESTEID-SK 2007 OCSP RESPONDER 2010 <span>OCSP/QC</span> <span>Withdrawn</span>
ESTEID-SK OCSP RESPONDER <span>OCSP/QC</span> <span>Withdrawn</span>	ESTEID-SK OCSP RESPONDER 2005 <span>OCSP/QC</span> <span>Withdrawn</span>
SK OCSP RESPONDER 2011 <span>OCSP/QC</span> <span>Granted</span>	SK Proxy OCSP Responder 2009 <span>OCSP/QC</span> <span>Withdrawn</span>

Figure 24. List of certificates issued to SK ID Solutions AS for the purposes of Qualified certificate for electronic signature

every couple of years. As of early 2022, four certificates are active, and all will be added to the trusted CA list.

In Estonia's case, SK ID Solutions manages the CA certificates [?]. Of the three certificates found on the EU Trust Services Dashboard, only two are relevant to us, the 2015 and 2018 ones, as the 2016 one has its purpose for use in Smart-ID, which the Web eID framework does not support.

The final count of certificates is six. Four certificates are required to support Lithuania and two for Estonia. It is essential to keep track of these certificates as each one of them can act as a point of compromise and must be monitored in the event they are revoked for security [?] or other issues.

**Exposing the service** With the certificate validation service configured, it is now required to link it to the Web API. If the company orients around using microservices, this service can be just that. All that the validation service requires is to expose an endpoint that accepts a nonce and a token from the javascript library and returns a validation result.

Companies must take proper measures to protect such service from adversaries as it acts as a fundamental trust anchor. Developers should take steps outlined in assume breach to mitigate the risk of misuse with the use of TLS and other countermeasures.

citation  
missing

## 8 Discussion

### 8.1 Do businesses even want eID?

When conducting the initial investigation on what criteria we should use to compare different eID providers, we interviewed the CTO of a logistics company. The full interview can be found in the appendix. The responses about current practices were shocking but not surprising.

**Authentication or Digital Signatures. What is more?** When asked if there was a choice between implementing eID authentication and qualified digital signature infrastructure, the company's focus would be on digital signature. Authentication only helps ensure the confidentiality and integrity of data and requires an additional heap of technological measures to prevent bypass. Qualified digital signatures offer an immediate benefit in the form of legally binding documents.

**Trust. What trust requirements the eID provider should fulfill for you to adopt it?** When asked if ISO/IEC 20001:2013 certification is sufficient, the answer was a resounding no. This certification should be the bare minimum for the company to consider using that solution. The CTO would "like to see that government portal, or banks are adopting this solution. This provides sufficient trust into the solution". This quote supports the assumption at the start that widespread adoption is low because currently, there are no big-name adopters outside of banks and governmental agencies.

**Source. Does the eID have to come from a TSP?** The company CTO was not concerned much about the kind of eID provider is: primary or third-party. As long as other large entities the solution is trusted by other entities (governments or large companies), there is no significant difference between choosing services from an eIDAS QTSP and not. This logistics company sees no clear advantage in creating a contract with SK ID Solutions to implement Smart-ID authentication over an agreement with the Estonian Internet Foundation.

One can argue that a TSP is the trusted solution by large governmental institutions; however, it comes with a problem, especially in Estonia, of poor market reach, which is also highly important.

**Market reach. How much impact does it have?** While trust and security in solution are the main deciding factors, increasing security would not attract companies to use an eID solution after it reaches a certain widespread adoption level. What will have more impact is the market reach.

We have presented the interviewee four options: eeID (eIDAS), Dokobit (private company), Web eID (DIY), Smart-ID (narrow specialty TSP). With all their advantages and disadvantages, assuming all reach the trust requirement and price of operations are similar, the CTO's option was the one with the highest market reach. The reasoning behind it was saving money on implementing multiple providers, and the less company has to do, the lower the risk of something going wrong.

In short, a larger market size would positively impact the decision process of choosing a particular eID provider.

**Pricing. How much is worth spending on eID solutions?** Reducing costs is one of the cornerstones of running a business. When presented with the ballpark of how much the company would have to spend to operate an eID solution, the company's CTO suggested looking at the broader market for identity management solutions. They currently use Azure for their services. The company would still need to pay Microsoft for their accounts to access the cloud platform infrastructure. Azure AD B2C is used for all other use cases, which provides identity management options the company is used to, and the operational cost is close to nothing.

The issue with the eID solutions is that they are targeting a different kind of company. Still, it is not clear which industry would willingly, without regulatory requirements, choose to implement such a system. The CTO estimates that for 10 000 authentications per month, a company could reasonably support 300-400 active users. This price effectively means adding 1€ per system user to operational costs. Not many industries can afford such a luxury.

**Technological hurdles** The system is only as secure as its weakest link. The hardest part of implementing an eID solution is not integrating with an external provider but creating access controls for new or existing resources. These measures will have to be in place for all interaction methods - from user interfaces to database and backup solutions.

This feels out of place, because it is, should I put it in its own chapter?

**Summary** Ultimately, the eID authentication solution suffers from a lack of benefits for companies. This solution deals with authentication and, by extension, access control. There is no visible advantage of using eID over a regular MFA solution from Microsoft. The company is just not dealing with data that would warrant that high level of assurance.

The overall tone of the interview is that for the eID authentication to be helpful, there should be a legitimate interest to obtain the user's national ID code. There are cheaper alternatives available for those interested in only the additional security measures.



## 8.2 Do businesses even want digital signatures?

If there is one quote to take from the interview, it must be "today [business owners] open PDF and apply PNG of signature into the file free of charge." Part of the reason why eID authentication is not widespread is that digital signatures are not widespread.

The benefits of eID in the private sector, even after the research, remain unclear. In contrast, the value provided by Qualified Electronic Signatures is obvious.

The only real business value eID authentication provides - trustworthy audit logs in the case of legal disputes. Still, even then, there are no high-profile cases in court on that matter. Although, it would be harder for defendants to claim they didn't access the system when logs clearly showed. Unfortunately, even that argument collapses when you look at the trust chain - people in power can sabotage AuthServer, issue tokens in the victim's name, fabricate access records.

The only thing that is legally binding is Qualified Digital Signatures. Only special approved devices can create digital signatures. There is no higher authority like AuthServer that can doctor these signatures.

Unfortunately, even with the visible advantages of electronic signatures, business owners still do not use them. Having a picture of a written signature inside a PDF document remains a popular way of doing business.

## 8.3 Which eID provider to choose?

The three case studies were not selected at random; they represent different approaches to accessing the electronic identity. Summary of pros and cons can be seen in table ??.

**Which to choose?** First, companies should decide if they need eID authentication in the first place. After that, it depends on priorities.

If the highest degree of trust factor is required, a company will have no other option other than to use a QTSP.

If high market reach and stability are required, adequately vetted and audited eIDAS node access is likely to be the best choice.

If the highest market reach is everything, adequately vetted third-party providers are a great choice.

In short, there is no clear advantage of one option over the other, and companies should address the options available to them individually.

Footnote on ID generation. Why not use the id and not store? Generally you would want to store who created a internal document, or who last edited a page. Using national ID for this purpose can open pandoras box when it comes to logging. Pseudonimization and linking the keys behind strong locks is next best thing.

Table 3. Advantages and disadvantages of each eID solution

Scheme	Examples	Advantages	Disadvantages
eIDAS	TARA, (eeID)	large target audience; officially supported and used by governments; cheaper than implement- ing many individual QTSP services;	may not include some more popular schemes; does not offer means to sign documents;
Third-party aggregator	Dokobit, Signicat, (eeID), (Web eID)	large target audience; includes schemes ex- cluded from eIDAS; cheaper than implement- ing many individual QTSP services;	not officially supported by governments or legisla- tion; trust issues; risk of provider ceasing op- erations;
QTSP	ID-Card, Mobile-ID, Smart-ID, (Web eID)	highest degree of trust; security audits are regu- lated by law; support for digital signa- tures; some options (ID card) can be free to operate;	very narrow market reach in comparison; complicated to integrate; operational costs can stack up quickly;

## 8.4 Dangers with having no control over identity

As per the case with using external identity providers such as Auth0, Azure AD, or AWS Cognito, companies put a significant amount of trust when using their services. These services create access tokens, which almost always contain some form of user-id, roles, or claims.

From a technical standpoint, nothing stops these companies from creating fake access tokens skipping the whole authentication process. As far as the relying party would be concerned, these tokens would be indistinguishable from real ones. A corrupt or compromised company would need to compromise only the last step of the authentication protocol - the one that sends (and optionally signs) personal information.

The same security concerns apply to state-issued electronic identities. We can identify three tiers of security, ordered from most to least secure:

1. Local device certificate authentication. Examples include ID cards and USB keys.
2. Remote device certificate authentication. Examples include Mobile-ID and Smart-ID.

### 3. Third-party authentication. Examples include eeID and Dokobit.

For example, to compromise a service relying on Dokobit, one would need to compromise any of the three listed services, as Dokobit depends on Smart-ID. With this logic, integrating Smart-ID directly is more secure as it removes a potential point of compromise.

**Is the attack likely?** Ultimately, governments should have no interest in compromising their systems. The only benefit of doing so is too contrived - they would gain the ability to frame someone in a heavily audited space to have an edge in court. There are more straightforward legal or technical methods to obtain supporting data about a person's activities.

More importantly, the issue with any system is that if an interaction method exists, one should not assume that only the intended users can access it. It is not unreasonable to think that backdoors can themselves have backdoors. Thus the only 100% safe countermeasure preventing anyone from accessing a system would be to make sure no one can.

In conclusion, it makes little sense for companies or governments to compromise their systems. The risks in doing so are astronomical in comparison to the benefits received.

## 8.5 Should one use eID for authentication?

Returning to the idea proposed in the thesis' introduction of having users register themselves with eID. After much research, the best solution was right under our noses the whole time.

**The core issue with eID authentication** In the thesis, we discussed only half of the data access flow. Secure authentication methods are essential, but one must not forget about authorization.

The eID authentication schemes are good at ensuring that someone is Alice with a high degree of certainty. All that certainty is not helpful if Alice is not allowed to access the resources at all. Deciding which resources Alice can access requires manual role assignment by a higher authority. Automatic assignments are a security hazard.

Consider an example. A company is hiring a senior developer Bob. They expect one to register soon, and yes, a person has registered. What should the computer do? Should it assign the role of a senior? Unlikely, as there are no guarantees that Bob was the one who just registered, it could have been Charlie, who is not a senior-level developer. If we only look at the data provided by eID, we will see that it was, with an incredibly high degree of certainty, Charlie, not a senior developer.

Alternatively, they could manually input the data, linking the account to a national identification code before they log in. This approach would work, but it would also defeat the whole premise of trying to skip manual verification. In short, manual confirmation is required when setting up authorization rules and would be impossible to avoid.

**A more straightforward solution** Instead of looking at electronic authentication, we can look at digital signatures instead. Consider the following flow:

1. Company employee creates a quarantined account for a new employee and gives a personalized registration link.
2. The potential client or employee enters there and is presented with a file they would need to sign. The document's contents are not of concern; it could be a privacy policy or terms of service.
3. After the employee signs the document, they upload it to the website. Employees can then verify that the name and surname in the digital signature match the person they were talking with online.
4. If the link given out in step one required them to register, this account could be taken out of quarantine, as enrollment identity was successfully verified.

This flow fully covers the initially proposed use case for eIDs, making them unnecessary. This approach is more convenient for the user and cheaper for the company. The only disadvantage is that it does not provide high assurance that the same person logged in or that the account was not compromised. However, companies can solve this issue by establishing internal operational security.

If it is still necessary to ensure high certainty after each authentication, the use of eIDs is always available.

## 9 Conclusion

In the thesis, we looked into the background, legality, and extensibility of eIDs, discussed the viability of using eID as a replacement for authentication, and analyzed three different eID providers the private sector could implement.

We have shown that it is possible to integrate an eID authentication scheme into a pre-existing SSO easily. The discovered challenges with integration are protecting the data from access by other means - no reason to force users to authenticate with eID if they can access the database with username and password.

We have created a privacy policy for the dummy test application considering the privacy requirements imposed by GDPR.

We have compared three different eID providers and discovered three different ways of performing cross-border authentication: integrating each provider individually, integrating a third-party provider aggregator, or tapping into a legally governed eIDAS framework.

We have discovered that all three authentication schemes have had security or integration issues in their protocols, limiting them in some tangible way.

We have interviewed a logistics company representative to gauge the acceptance of eIDs in the general public, only to discover that it is unlikely that companies will adopt this technology without drastic changes in the market.

We have proposed an alternative to eID authentication using a similar scheme when a high certainty behind a given identity is required during enrollment only.

**Summary** The ability to integrate eID in the private sector exists; however, the public acceptance of them is limited. The technical and legal challenges associated with eID authentication make it impractical to implement for almost all companies.

## **Appendix**

### **I. Glossary**

## II. Licence

### Non-exclusive licence to reproduce thesis and make thesis public

I, **Gediminas Milašius**,  
(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,

**Exploring integration complexity of different multi-national eID authentication solutions in the EU private sector**,  
(title of thesis)

supervised by Arnis Paršovs.  
(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.

Gediminas Milašius  
**11.06.2022**

### III. Questionnaire

The interview's goal is to understand better the reasons for the poor adoption of eIDs in the private sector. This interview was conducted with the CTO of a multinational logistics company.

1. With electronic key cards, users can authenticate themselves using a piece of hardware, say a card or a USB stick. This authentication method is often more secure than the usual username and password approach. Are you and the company in general aware of this?
2. I would describe an eID scheme as something like your id card, but digitally. There are three main schemes in Estonia: ID cards, Mobile-ID, and Smart-ID. Are you familiar with at least one of them?
3. With the eIDAS regulation, these three eID schemes can create digital signatures, with the legal value of a handwritten signature. Do you have a place in the company where you print a document, sign it, scan it and upload it? Would you switch to a solution that would avoid this process?
4. Without disclosing the worth of transactions floating around the company, would the security benefits of the eID schemes benefit company enough for you to switch to using them?
5. Authentication and signing usually come hand in hand, but if you were to have the ability to choose, assuming authentication and signing both costs equally as much to implement, would you rather spend the resources on authentication or digital signing? SEB Bank used to or still allows for transactions under 50€ to be done without signatures, only authentication. Would you at that point no longer consider the authentication method entirely?
6. Your company deals a lot with automation. Would you be comfortable automating the use of digital signatures in your company's name, or would you rather still have a person at the end manually reviewing and signing documents?
7. Say a human mistake occurs: a person mistakenly signs a document they shouldn't have, and the company faces losses. It would be easy to track who made a mistake with digital signatures. What would your company do in that case?
8. I have four different authentication options a company can take. Assume you would have to pick one of the four and explain the main reasons for your choice.  
The first option uses the primary eIDAS network of Europe to authenticate themselves to any EU public sector service. For example, a Lithuanian citizen can use



their eID to sign into Estonia's banks. This network's security is held to the highest standards. Some discrepancies appear because of the criteria, such as Estonians being unable to sign in via Smart-ID to foreign websites. It is significant as a lot of people use Smart-ID. Do you think it is an acceptable solution for you?

The second option would use a company in the middle whose sole responsibility would be to federate the sign-in process. Like the first authentication method, you can also sign in from many more European countries, but this time without using the eIDAS network. A clear advantage over the first one is the more lax security requirements, allowing other authentication methods such as Smart-ID. Keep in mind that this authentication method is still highly trustworthy. Would you consider the ability to reach a broader audience at the cost of not using the official infrastructure a risk worth taking?

The third option puts a lot more risk on the company and allows for only a narrow market band. I am talking about smart cards and how a company could accept one, but the server should never trust the certificate a card sends. This approach is challenging to integrate and susceptible to many attacks; however, its advantage is that it is free to operate. If we ignore the personnel costs for maintaining the trust certificates, that is. Would no operational fees be convincing enough to pick this option?

The last option is similar to the third about the challenging implementations and the narrow market band. This time you will not have the advantage of free operational costs. However, you will still benefit from not having an intermediary company. This option would be if you integrated with Smart-ID directly. Is having an intermediary company of concern to you?

9. What is an acceptable price for a single successful authentication? The business model of options 1, 2, and 4 is to charge an amount per authentication. Let's aim for around 10 000 authentications per month; how much do you think is acceptable to spend on such a number? Would 500€ per month be acceptable?
10. Options 2-4 also create digital signatures; the first cannot. Does your opinion change at all about which solution you would pick?
11. An alternative to using government-issued eID solutions, you can also issue them yourself at a highly reduced price and trust factor. This solution is still more secure than the username+password approach. If you were to change how the company performs authentication, would you switch to the internal system, eID scheme, or not switch at all, and why?

## IV. Privacy Policy

1. This document explains which personal data is processed on the AuthServer website (auth.eid.gedas.dev) for use as part of master's thesis research.
2. Information we process:
  - (a) User's eID authentication data, which can include: given name, surname, country of eID issue, unique identifier provided by eID, birth date;
3. Information we store:
  - (a) registration email address as part of the account creation process;
  - (b) user's country and unique identifier as provider by services used for external authentication;
4. Data retention policy:
  - (a) all data is wiped from the application at 00:00, Estonia time;
  - (b) users can manually remove their data by visiting <https://auth.eid.gedas.dev/Identity/Account/Manage/PersonalData>; effective immediately;
  - (c) users can download their personal data on the same page as (b);
5. Data shared with third-parties:
  - (a) information received from Dokobit service is subject to UAB Dokobit privacy policy;
  - (b) information received from Web eID service is subject to RIA's privacy policy;
  - (c) information received from eeID service is subject to internet.ee privacy policy;
  - (d) when checking the validity of certificates, the issuer defined in the certificate received will receive a certificate identifying information to validate the revocation status;