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

Gediminas Milašius

Integration analysis of various eID authentication solutions used in the private sector of Estonia

Master's Thesis (24 ECTS)

Supervisor(s): Arnis Paršovs, PhD

Tartu 2022

Integration analysis of various eID authentication solutions used in the private sector of 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 go to banks and e-government services. If any other business in the private sector 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 available eID solutions and providing security and integration analysis for Web eID, eeID, and Dokobit.
The main findings of the thesis show that the technology to support eID authentication exists and that most businesses choose not to use eID authentication because the benefits of using such a system do not outweigh the costs of integration. Additionally, this thesis discovered significant security vulnerabilities in some eID solutions, previously assumed to be safe and secure.
The thesis results serve as a reminder not to assume that a product is secure just because it specializes in security.

Keywords:
eID, Authentication, eIDAS, eeID, Dokobit, Web eID, Estonia, EU, private sector

CERCS: P170 (Computer science, numerical analysis, systems, control)

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 [1], 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.

There is another use case for using this form of identity establishment. 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 [2]. 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's 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 [3]. 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 level of assurance, from substantial to high, over the certainty of the authenticated person's identity. This technology allows for verifying a person's 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. Unknown risks are an excellent deterrent to innovation, making companies reluctant to use new technologies. This research may find it favorable for companies to take risks associated with implementing the new technology and start 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 the steps they would need to take to adopt the technology. From this goal, we extracted the following research question:

What is the best eID authentication solution available to an Estonian EU targeting enterprise?

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

1. What are the prerequisites for a given architecture to be able to support eID solutions?
2. What are the different eID authentication solutions available to Estonia's private sector?
3. How do various eID providers compare based on the following questions:
   (a) How trustworthy is it to process sensitive information?
   (b) How large is its market reach (in countries)?
   (c) How expensive is it to operate?
   (d) Does it inconvenience the end users any more than the regular eID providers would?
   (e) How complicated is it to integrate?
   (f) How complicated is it to maintain?
   (g) Is the eID authentication solution provider protected against common protocol attacks?
   (h) Is the integration manual comprehensive enough to protect the relying company against common protocol attacks?

From the initial question, the word "best" is ambiguous. This part was further narrowed down into six questions (3a-h) to help clarify it. These criteria were chosen based on the feedback provided by a CTO of a logistics company. We provide the full interview in the appendix I.

Hypothesis To help answer the last four questions, which have no answers in literature, we have created the following hypotheses:

1. The eID solution providers give clear integration instructions.
2. The eID solution providers give clear hardening instructions.

3. The eID solution providers themselves are not vulnerable to common protocol attacks.

4. The eID solution hardening instructions cover all common protocol attacks.

We will verify these hypotheses for each of the eID solutions.

1.3 Research Method

The previous section outlined the questions we aim to answer in this thesis. This section will describe how we will obtain the data necessary to answer the questions.

Question 1: What are the prerequisites for a given architecture to be able to support eID solutions?

This question aims to identify which systems are capable of supporting eID solution integration. The validation can be done with a simple experiment of integrating any eID solution. If successful the tried system is capable of supporting eID.

Question 2: What are the different eID authentication solutions available to Estonia's private sector?

This question aims to enumerate already existing solutions the private could integrate today. To our knowledge, there is no compiled list of solutions available.

To answer this question, we will construct a list from the various scattered sources of information. The final list is likely not to be exhaustive.

Question 3: How do various eID providers compare?

Each of the following questions will be answered on a case study basis by analyzing some eID solutions in-depth.

Question 3a: How trustworthy is it to process sensitive information?

Trust is ultimately a subjective topic. To help companies make an informed decision, we will look into who the provider is (e.g., a government institution or a private company) and their background.

All information required to answer this question will be obtained from existing literature.

Question 3bc: How large is its market reach? How expensive is it to operate?

The company interested in integrating an eID solution is a for-profit organization, so it is in their interest to know what audience they can reach and how expensive it is to do so. Fortunately, the analyzed eID solution providers are also for-profit and list their price

sheets publicly. Market reach is a significant selling point for them, so that information is also available.

The information existing in the literature is sufficient to answer these two questions.

Question 3d: Does it inconvenience the end users any more than the regular eID providers would?

Our company wishes not to inconvenience the users any more than they need to, so if there are additional hurdles users would need to overcome when accessing a website (such as requiring to install additional software), the company would like to avoid them.

In our case, "additional software" does not include software used to access the government services of the issuer country. For example, a software solution distributed by the Spanish government and capable of accessing Spanish public services using Spanish ID cards does not fall under additional software. In contrast, if the solution was created and distributed by Estonia and is not capable of accessing Spanish public services using Spanish cards will fall under additional software.

This information to answer this question would be obtained by integrating a solution and verifying it works or from literature.

Question 3e: How complicated is it to integrate?

This question aims to answer how many development resources a company would need to invest in adding users' ability to log in with their eID. We also ask how difficult the solution would be to maintain after the active development process has finished.

An intuitive metric to measure integration complexity would be to track the time taken to finish integration for each solution. The main issue with this approach is that it favors the solutions integrated last, as the developer would obtain the necessary skills to solve issues from prior solutions. Giving this task to multiple developers to work on is also not ideal as the skill level between them varies too greatly.

Instead, the measure we have chosen is to track the inconsistencies between the actual data provided by the eID solution provider and their provided documentation. In the best-case scenario, there will be no ambiguity or contradictions. The higher the number of inconsistencies, the lower the score for integration.

This question relates to hypotheses 1 and 2.

Question 3f: How complicated is it to maintain?

The maintenance part of this question will track the amount of moving parts the client company must add to support the solution. The more significant the quantity and the more complicated they are, the lower the maintainability score will be.

Question 3g: Is the eID authentication solution provider protected against common protocol attacks?

Each of the eID authentication solutions will be subjected to the Internet Engineering Task Force (IETF) guide on the best practices to protect against attacks on the OAuth2.0 [4]. OAuth2.0 [5] is a popular federated authentication framework. While the analyzed eID solutions may not necessarily use OAuth2.0, all HTTP and browser-based attacks have similar issues, and it is easy to draw parallels between the authentication frameworks.

The IETF guide provides a list of attacks and countermeasures against these attacks, and the eID authentication provider must be resilient against all of them. Failure to meet all of the criteria would deem the solution unsafe to use.

This question relates to hypothesis 3.

Question 3e: Is the integration manual comprehensive enough to protect the relying company against common protocol attacks?

For an authentication protocol to be secure, both the identity provider and relying party must take measures to harden it. To answer whether the manual provides sufficient instructions to mitigate attacks, we will integrate the eID solution with the provided specifications and verify if the final result is resilient against attacks by using the same IETF document as used to answer question 3g.

Failure to provide sufficient instructions to mitigate against all attacks would negatively impact the score for integration.

This question relates to hypothesis 4.

1.4 Scope

To not cover every possible scenario, we will be making a couple of 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) When analyzing an eID solution for integration complexity, we will only consider using an HTTP-based SSO. We chose to support only a single local identity provider to eliminate as many variables as possible, as all analyzed solutions would have to fit into a similar structure.

Company is committed to getting some form of eID authentication system in place This means they did the market research, and management found it favorable to invest in eID authentication. Analyzing which companies would benefit from eID authentication or if they should invest in the first place is outside the thesis's scope.

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

1.5 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 solution providers can take for integrating cross-border authentication;
4. assessment of various data transfer protocols in use for eID authentication;
5. security assessment and disclosure of weaknesses in analyzed eID authentication solutions;
6. collaboration with the Estonian Internet Foundation to help develop their eID solution;
7. collaboration with UAB Dokobit to help fix a discovered vulnerability in their eID solution;
8. release of a source code example on integrating eeID, Dokobit, and Web eID solutions.

1.6 Structure of work

yes

The thesis will consist of the following main chapters:

Section 2: Background This chapter contains literature relevant to understanding the terminology and concepts used later in the thesis. Additionally, it contains a list of currently available eID providers in Estonia alongside a short description.

Section 3: Related Work This chapter references literature that covers similar topics to the ones covered by the thesis. These studies may be done with different technologies or in different countries.

Section 4: Architecture Definition When integrating an eID provider into an existing system, one must first know how the system is composed. Here we provide a schema and process overview and address inherent weaknesses in the base system.

Section 5-7: Case Studies These sections look at data flow, trust, pricing, security requirements, integration specifics, and discovered weaknesses for each provider in the thesis (eeID, Dokobit, Web eID). These sections spotlight the advantages and disadvantages of each provider.

Section 8: Discussion This section summarizes the findings from the previous three chapters, provides a comparison between the three eID providers and draws a conclusion about which is the best solution for different use cases.

2 Background

2.1 Electronic identity (eID)

In Estonia, digital identity has been around for over 20 years [6]. 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 [7]. 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 [8]. 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.

To better illustrate the differences, we will look at two countries with very similar eID schemes: Estonia and Lithuania.

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 [9]. 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, like PaySera, are listed on the public sector gateway portal and raise 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 [10] from use in TARA, a service that all primary public services rely on.

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

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

2.2 eIDAS

The eIDAS regulation [3] provided the groundwork for recognizing the signatures issued by other EU countries by imposing strict liability and mutual-recognition requirements. 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 in such a decentralized manner. Failure to validate certificate signatures could have severe consequences [11]. 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 [12]. 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 [3]. 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 [13]. Smart-ID is not a scheme notified by any of the EU member states.

In Lithuania's case, only the smart card solution is allowed - no mobile log-in methods have been notified [13].

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 [14] 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 [15]. Over time it saw an additional use case - a secure and trustworthy authentication method for the public and private services [16].

Researchers found that the bank link protocol, while applicable, was in some cases "extremely insecure" [17]. From March of 2021, RIA has disabled the use of bank link to access public services [10], which accounted for only 1 percent of all authentications, revealing its overall popularity.

2.3.2 Smart card (ID card)

ID cards are one of the most common ways for citizens to access their eID in Estonia, primarily due to the legal requirement of owning an identity document. The Identity Documents Act [6] 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 [18, 19]. Smart card functionality touches not only ID cards but residence permits and other government-issued cards as well [13].

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) [20] 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 [18]. This certificate, once on the computer, can be sent to the private company's authorization server with Client Certificate TLS option [21] or with the use of a specialized helper library, using standard REST calls [22].

Qualified trust service provider for Qualified Certificates for e-signatures installs the certificates in ID cards [23], 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

In 2007, 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 [24]. 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 relying party 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 [25]. For the end-user, mobile operators can charge an additional fee for the use of this service [26].

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 [23], 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 three of the Baltic States [27]. The protocol utilizes mobile phones as authentication, similar to Mobile-ID. Unlike Mobile-ID, however, it does not require specialized external hardware [28]. 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 [29]. After passing the audit, Smart-ID was recognized as a QSCD, allowing it to create QES in 2018 [30].

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 [31]. For users, unlike Mobile-ID [26], 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 uses their data centers to hold part of the private key and certificate used to authenticate users [23], 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 [32]. TARA provides the ability for users to log 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 [32].

Unfortunately for private businesses, the Estonian Information System Authority intends to limit the use of TARA to public services only [33].

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 [34, 35].

It is worth mentioning that while the underlying authentication methods have received proper eIDAS auditing and are backed by a qualified trust service, all data sent from a third party no longer carry the same degree of certainty.

2.3.6 eeID

Estonian Internet Foundation created the eeID service for the exclusive purpose of bringing eID authentication to the private sector [36]. It is a clone of TARA but without access to 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 [37].

The vision of the eeID service is to allow users to access the markets of all EU countries. Unfortunately, the cross-border authentication feature is disabled as of the time of writing.

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 [38]. ID cards are only legally required to be held by citizens over the age of 15 [6], 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 provided exclusively for the education sector.

2.3.8 Dokobit

In the initial list of services using eID in Estonia [14], one service stands out - Dokobit [39]. They provide services comparable to eeID in that they aggregate different eID methods used 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 [36].

Pricing for Dokobit varies drastically, and the provided prices for the Baltic States [40] 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 [39].

The company UAB Dokobit has been given certification for being a trust service provider for Qualified validation of qualified e-signature. It means the service itself does not provide Digital Signature certificates but was properly audited under eIDAS for trustworthy verification of signatures [23].

2.4 Confidence over someone's identity

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 [3]. These assurance levels map to levels 2-4 of ISO 29115 [41]. The supported schemes we will analyze in the thesis will all have a rating of at least substantial, making them provide a high degree of confidence over someone's identity.

To illustrate the differences between the levels of assurance, we can create an example. 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) [42].

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 levels.

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 goal 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 perform a risk analysis for the data they are protecting and then analyze what level of assurance they would need.

2.5 GDPR

Two years after the EU enacted the eIDAS legislation, the EU parliament issued new legislation, General Data Protection Regulation (GDPR), to consolidate all previous privacy laws [43]. It is essential to be aware of it, as when dealing with eID, personal data processing is unavoidable.

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 essential concepts.

Personal data The eID solutions in the scope of the thesis all provide the following personal information: full name, serial number, and country of origin. This serial number can be any code that would link to a person. In Estonia, the serial number is the national identity code.

Processing The minimal amount of data processing required for authentication is the storage of 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 most basic case, no third parties are involved, and the controller, the processor, and the recipient are the same entity - the company.

2.6 Threats

Each of the eID solutions uses some form of medium to transfer information. This medium is not impenetrable, and companies should be aware of the threats they would need to address before integrating an eID solution.

The sequence diagram (see Figure 1), shows a general data flow 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).

Trusted communications commonly operate using an encrypted backchannel, whereas untrusted ones require the client to send plaintext data from the eID provider. Because an 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.

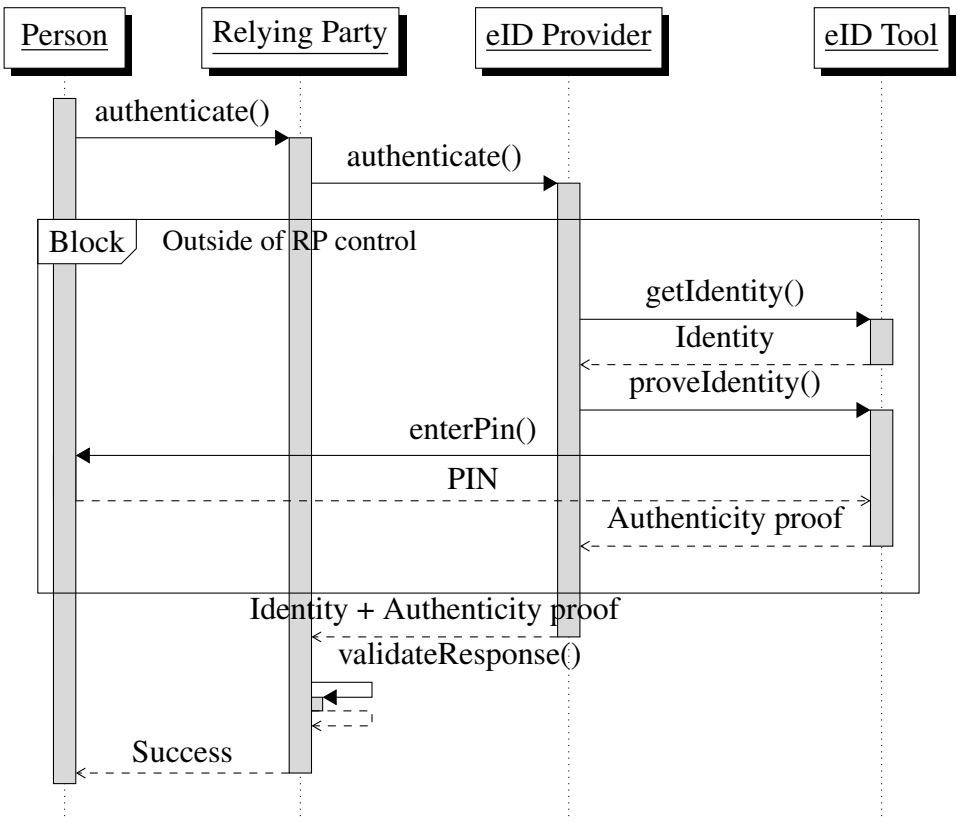


Figure 1. High-level sequence diagram of a eID authentication flow

3 Related Work

3.1 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 [44]. 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 [44, 45].

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.

3.2 LEPS - Leveraging eID in the private sector

This final research [46] was performed at a similar time to the eID@cloud [47], 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."

3.3 Federated Identity Architecture of the European eID System

The authors of this paper describe the current situation in the identity management landscape [48]. The researchers provide all the necessary background information to understand the implementation details of any eID authentication system design.

3.3.1 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) [49]. 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.

3.3.2 Authentication Paradigms and Models

The second helpful point of the paper is the description of different identity management paradigms and models [50]. Paradigms refer to the implementation and deployment, whereas models refer to the data storage and roles.

The three main paradigms as 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 [6, 36, 39].

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.

3.3.3 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 [51], OAuth2.0 [5], and OpenID Connect [35] 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.

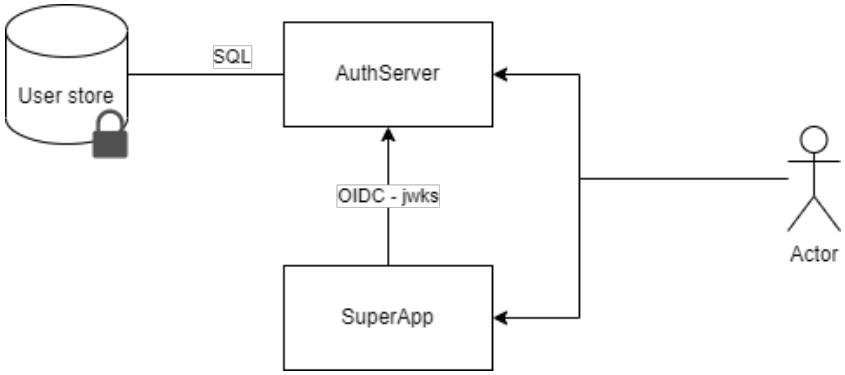


Figure 2. Initial system architecture

4 Architecture Definition

In this section we will outline the initial state of the system, before the introduction of any eID solutions. This sets a baseline to accurately test and validate the findings of the research.

For better clarity with terminology, the company wishing to integrate eID authentication is a physical exercise data tracking enterprise, **WorkAuth**. They are the relying party with respect to the eID solution providers.

4.1 System architecture

This section will focus on the technological architecture surrounding the integration; we will explain the business requirements for the system to better illustrate the intended use case.

4.1.1 System overview

Initial state Figure 2 shows us 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 authentication server. 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 in the company.

2. SuperApp. A resource server with access control enabled. It uses OIDC id tokens issued by AuthServer and verifies them using public key cryptography. 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).

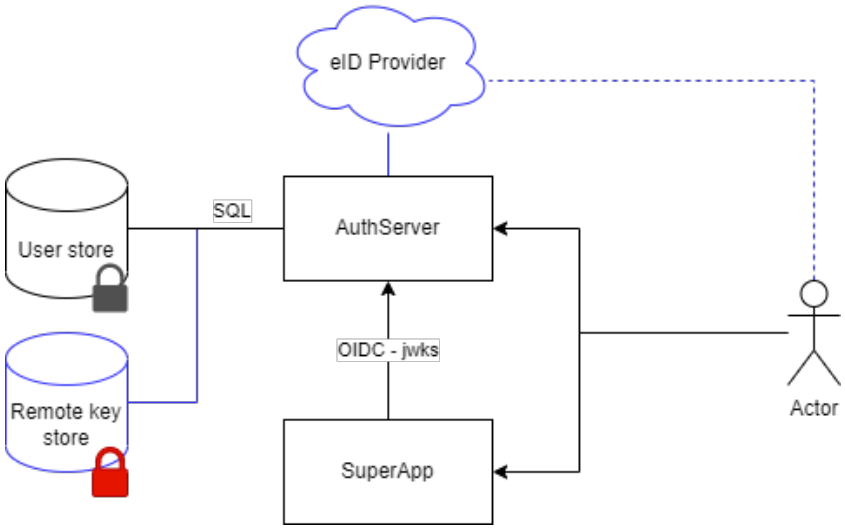


Figure 3. System architecture after the inclusion of an eID provider

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

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 3. 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. These keys are the person identifying information received from the eID providers.

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 [52], companies should foresee cases where numerous eIDs could map to a single internal ID.

In the diagram (figure 3), the key store has a red lock next to its icon. The data stored there may be subject to more strict privacy regulations in some countries than others.

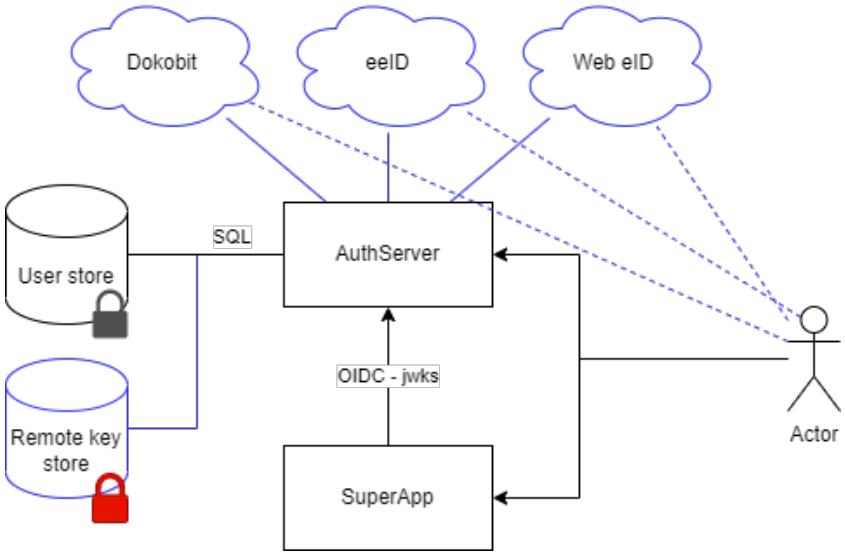


Figure 4. System architecture after the inclusion of all eID providers in scope

Companies must ensure sufficiently strict access control for this part of the infrastructure to satisfy the legal requirements.

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.

4.1.2 Process overview

The desired system must satisfy two business use cases.

The first o (see Figure 5) is concerned about accessing a protected resource with a token issued by the AuthServer. This use case is part of the base state of the system and validates that the baseline implementation is correct.

The second use case (see figure 6) is also concerned about accessing a protected resource, but to limit access to only those, who authenticate with a higher level of assurance, like an eID solution. This process validates the successful implementation of eID authentication and access control.

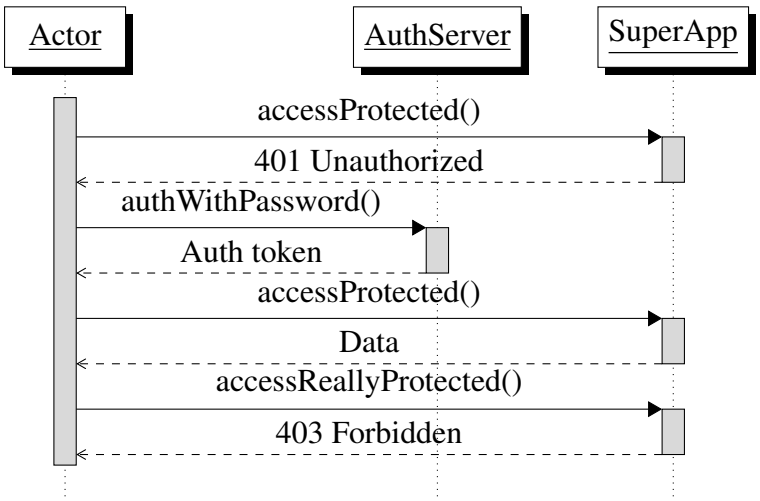


Figure 5. System behavior when authenticated with a username + password scheme

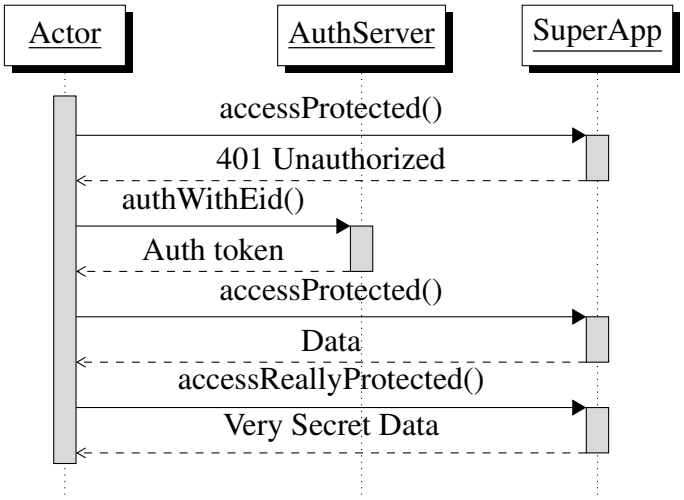


Figure 6. System behavior when authenticated with an eID scheme

4.1.3 Linking eID to an internal user ID

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.

Using Estonia's passport as an example, 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, another EU member state'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.

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 from the country. Each country must provide an identifier, even if the country doesn't have assigned a code to a person [52]. How countries achieve uniqueness is not of network's 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" [52]. 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 z'' company's case. To uniquely identify a person, we will only need its origin country and a unique identifier a member state must provide.

As a result our chosen identifier structure will be of the following: "{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 [43]. Before going live with an eID solution, they must conduct a legal audit for compliance.

A privacy policy is a legal document and is 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 [53] and Dokobit [54].

From the cursory analysis of the two policies, there are three fundamental aspects any 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 II 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 data 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 data transfer 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 3, four system components do not interact with an eID solution: AuthServer, SuperApp, application store, user store, and remote key store. We can identify weaknesses in those parts and analyze each of them through the lens of CIA (confidentiality, integrity, availability) security analysis.

4.2.1 SuperApp

[CI] Users can see and or edit data normally forbidden to access This issue is caused 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 protection on the API endpoint.

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 [35] 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 this vulnerability, 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.

In short, if a person logs in with a low level of assurance, they should not be able to add a higher level of assurance authentication scheme to the same account as that would artificially elevate their account trust.

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 [55]. 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 [56].

[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

The Estonian Information System Authority has created TARA - a gateway for public sector services to integrate eID authentication easier and cheaper [33]. It is heavily inspired by the OpenID Connect [34] 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 [36].

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 [13]. Currently notified countries (in order of time notified) include Germany, Italy, Croatia, Estonia, Spain, Luxembourg, Belgium, Portugal, Czech Republic, 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.

5.1 Authentication Protocol

The underlying data transfer protocol used by eeID is almost identical to OpenID Connect code flow [34, 35]. One irregularity exists between the OIDC spec and eeID, which makes the whole flow non-compliant to the spec [57], which we will get back to in a later chapter. Otherwise, the flow is identical. A high-level overview of this flow can be seen in figure 7. It consists of seven main steps:

1. Initial log-in. Preparation for redirect to the eID provider and generation of secrets.
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. 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.

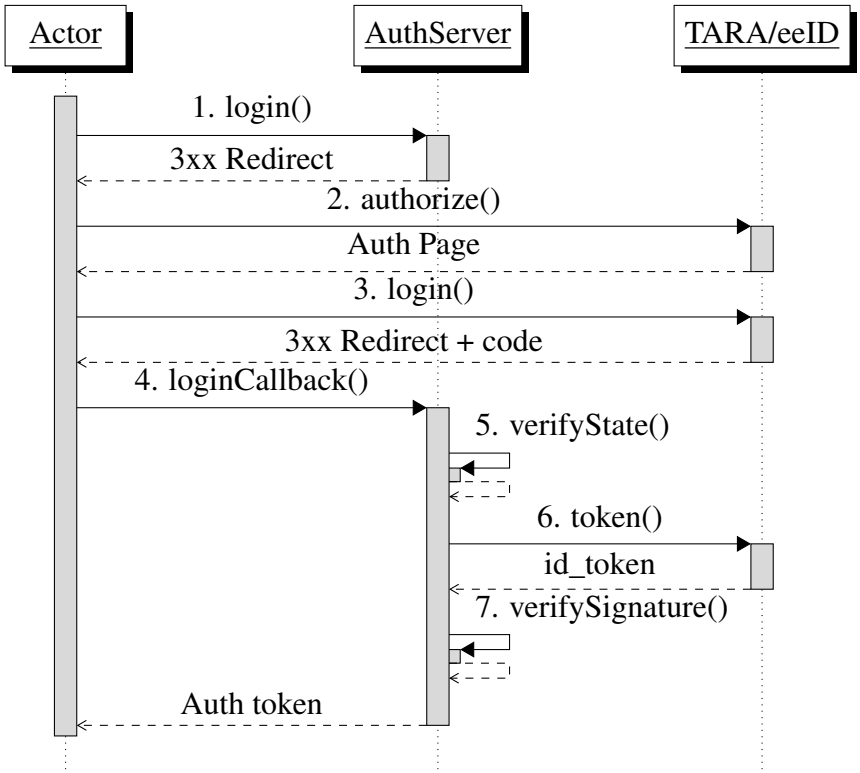


Figure 7. OIDC code flow used in TARA/eeID

5.2 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 identity data to create a standardized 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 inclusion of eeID service, another entity is added between them who has full read and write access to the data.

Integrity issues are caused by having that full access to the identity 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.

5.3 Pricing

The operational cost when using the eeID identity provider is nine cents per successful authentication [37]. It is the highest price among all identity providers and does not have tiers. A thousand successful authentications would cost WorkAuth 90 €.

5.4 Security Requirements

The TARA/eeID documentation has an in-depth integration guide and the validation requirements clients must implement [34]. 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 [34] and IETF's OAuth 2.0 Security Best Current Practice document [4].

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 causing any validation failures later on.

5.4.1 Protocol's built-in security features

OpenID Connect specifies three different flows - code, implicit, and hybrid [35]. In code flow, all sensitive tokens are handled via backchannel - a secure communications channel between the eID solution provider and the relying party where the user client is not involved. The implicit flow is the opposite - a user agent receives the identity token from the eID solution provider and sends it directly to the company's authentication server, which validates the the authenticity of this token. A hybrid flow is a mixture of the two. Security experts consider the code flow to be the safest option of the three [4]. Coincidentally, it is the only supported flow by TARA and eeID [34].

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 code 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 solution provider are sufficient.

Insufficient Redirect URI Validation

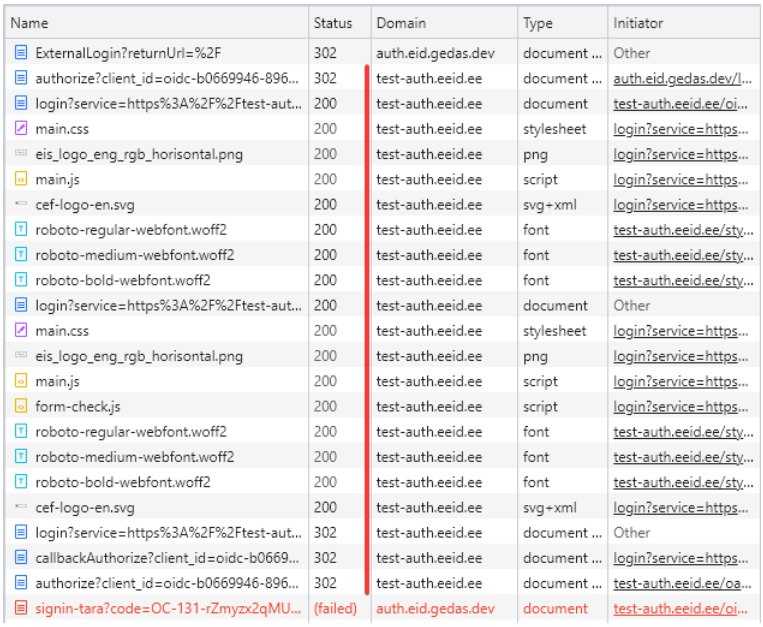
- Changes in the registered OIDC application undergo manual verification by eeID employees and do not allow for wildcards.

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 8). No mitigation measures are required.



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/oi...
main.css	200	test-auth.eeid.ee	stylesheet	login?service=https...
eis_logo_eng_rgb_horisontal.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/sty...
roboto-medium-webfont.woff2	200	test-auth.eeid.ee	font	test-auth.eeid.ee/sty...
roboto-bold-webfont.woff2	200	test-auth.eeid.ee	font	test-auth.eeid.ee/sty...
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_horisontal.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/sty...
roboto-medium-webfont.woff2	200	test-auth.eeid.ee	font	test-auth.eeid.ee/sty...
roboto-bold-webfont.woff2	200	test-auth.eeid.ee	font	test-auth.eeid.ee/sty...
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/oi...

Figure 8. The eeID service does not use resources outside of their domain in the authentication flow

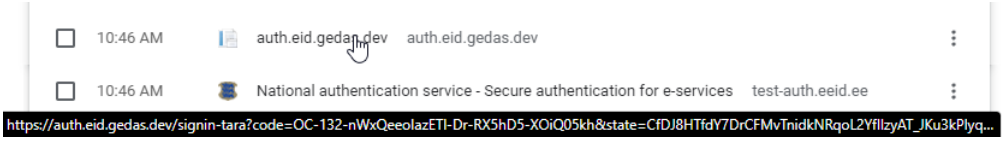


Figure 9. Authorization code is leaked inside the browser history when using eeID authentication

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 9); 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.

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

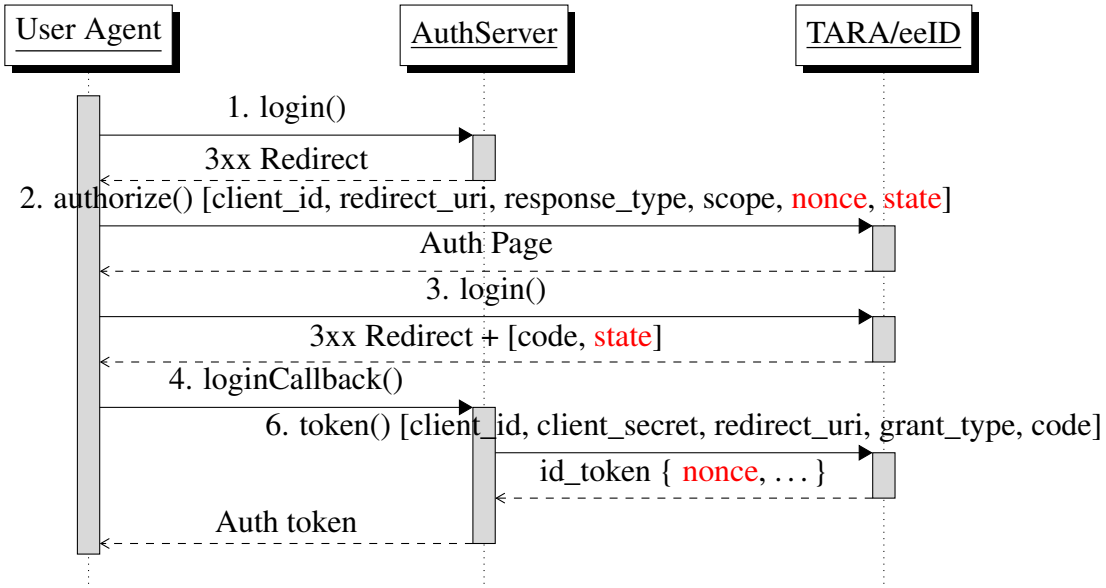


Figure 10. CSRF/Code injection mitigation in OIDC code flow

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 10 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 11).
- Relying party should integrate this or a similar countermeasure.

Mitigation measures are sufficient on almost all browsers released in the last ten years [58].

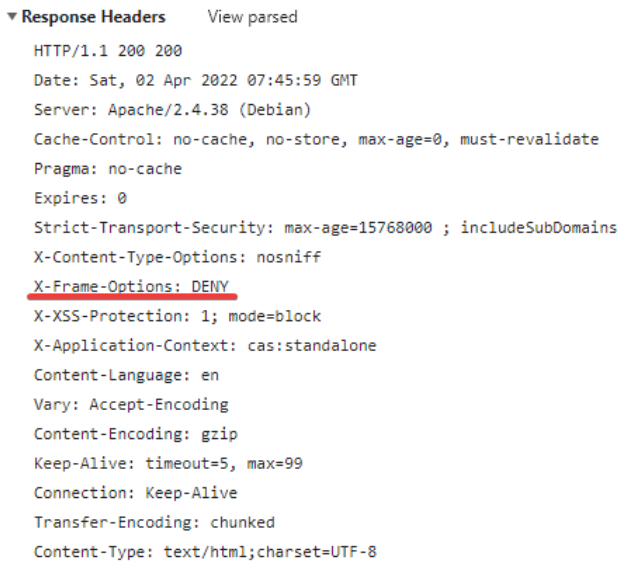


Figure 11. Response headers browsers receive when opening the eeID service's login page

5.4.2 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 10. 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 10), 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.

In step 6 of figure 10, 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

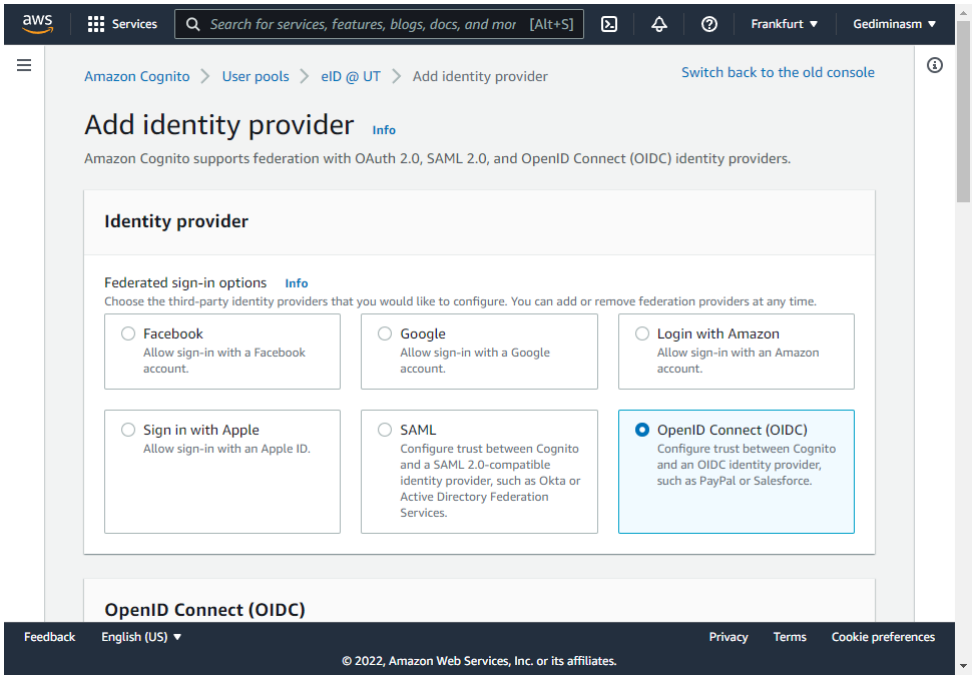


Figure 12. Adding OpenID Connect provider to Amazon Web Services

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.

5.5 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 12, 13). Alternatively, there are many officially certified services and libraries developers can use [59].

This thesis will use Microsoft's official ASP.NET Core library [60].

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

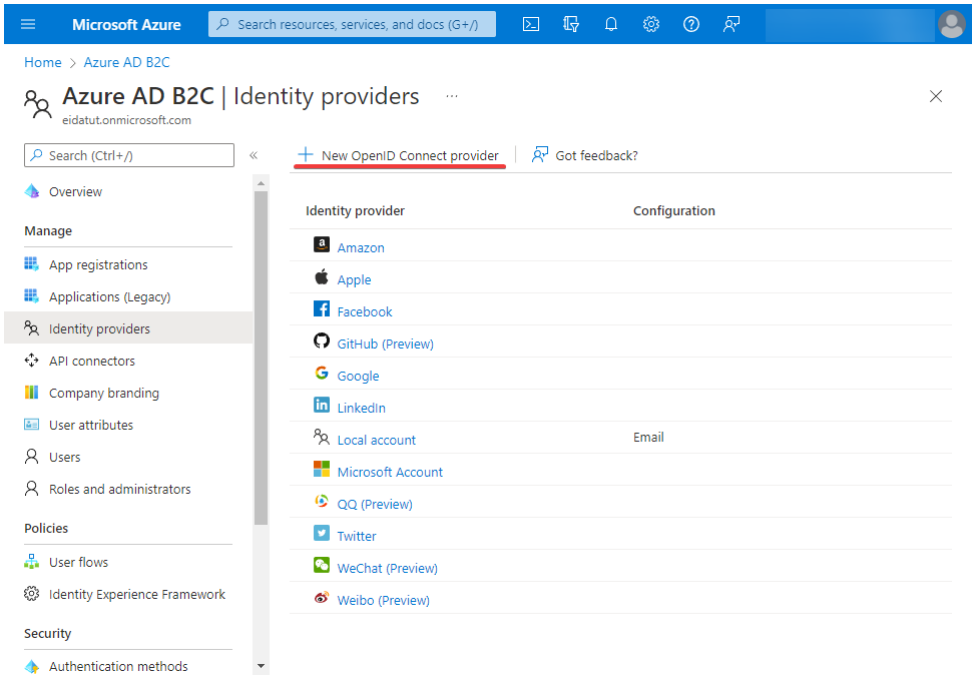


Figure 13. Adding OpenID Connect provider to Microsoft Azure

code from the .NET library [60] and the TARA documentation [34]. The steps listed will be as seen in figure 7.

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 [34]. 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 [4]. 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 1).

```
GET https://test-auth.eeid.ee/oidc/authorize?

client_id=oidc-b0669946-896b-4cdf-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 2). Note the inclusion of nonce, as this value should not be here [61] (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=J5BpaPNynnbhZCWmDlCZc5QWznVyIfebYGkZ3...
```

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 3 for a request example.

```
POST /oidc/token HTTP/1.1
Host: tara.ria.ee
Content-Type: application/x-www-form-urlencoded
Authorization: Basic b2lkYy1iMDY2OTk0Ni04OTZiLTRjZGYtYTQ3OC1hN-
    jBhZmQ1YzE4YTUtMjA6aHR0cHM6Ly95b3V0dS5iZS9kUXc0dzlXZ1hjUQ==

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) [5] 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 [34].

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.

5.6 Discovered Weaknesses

5.6.1 Incorrect implementation of at_hash

In the TARA Technical Specification [34], the identity token has at_hash value that is not according to the OIDC spec [35].

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 [62] 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 4).

```
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-
    VU6Y2LZjrNrVCdhlEaCxG6Gpzt0RsE-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 5). The implementation expects a different string than was provided. The same transcoding behavior is seen on the TARA Demo Client [62].

```
user@localhost:~$ access_token="AT-95-
    VU6Y2LZjrNrVCdhlEaCxG6Gpzt0RsE-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 [57].

5.6.2 Confusing state and nonce behavior

OpenID Connect specification does not mention 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" [5]. The primary security feature is to prevent CSRF attacks [5, 4].

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 [35]. 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 [63]. The only issue is that these tokens expire after 40 seconds and cannot be refreshed [34], making this approach impractical.

If we look at the data flow diagram (see figure 14), 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 [64]. 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.

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

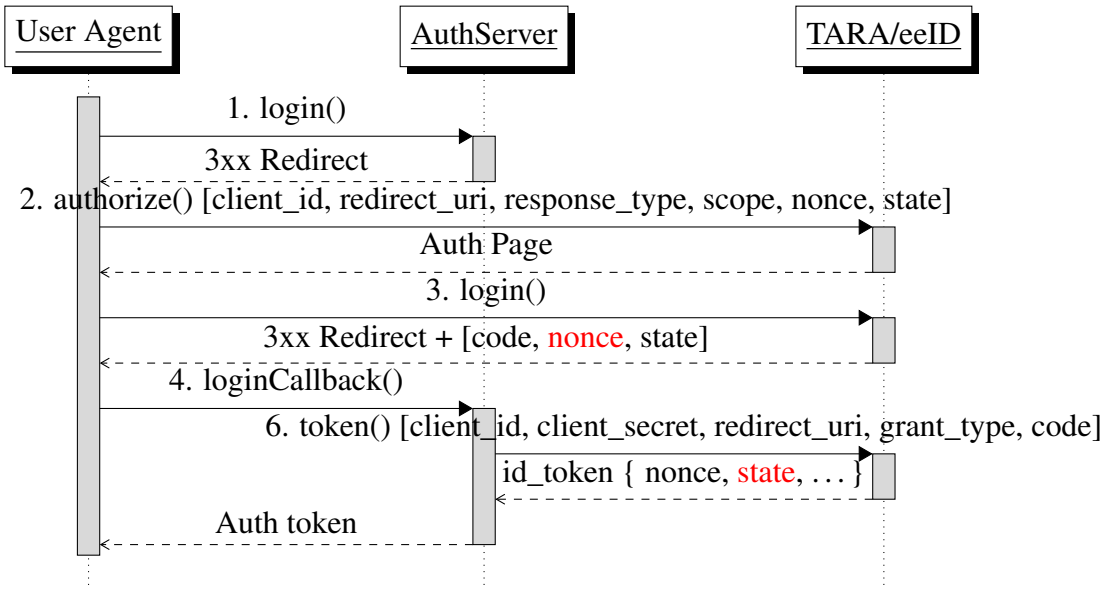


Figure 14. The incorrect OIDC code flow used in TARA/eeID

6 Case Study: Dokobit

Dokobit [39], trademark and subsidiary of Estina [65], offers two products: Document Signing Portal and API solutions.

The first product, the Document Signing Portal, was released in 2014 [66]. The primary purpose of this solution is to allow users to upload documents and digitally sign them online.

The second product, API solutions, targets businesses in a variety of scopes: signature collection, signing, identification, sealing, and TSP monitoring [39]. 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 [39] users to authenticate themselves with their countries' scheme.

The company has received ISO/IEC 27001:2013 certification [67] and in 2020 was included in the EU Trusted Service List [23, 66], 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 is that the Gateway uses a prebuilt UI hosted on Dokobit servers. 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" [68]. 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 user flows: embedded and redirect-based.

In embedded flow, users could stay on the website, authenticate in a pop-up window or frame, and update the website view accordingly after finishing the authentication process. This 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, similarly to OAuth2.0.

Ultimately, experts consider the embedded flow to be the weaker of the two methods [69] 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" [70]. This practice means that companies who have a mobile app or would consider having one in the future mustn't use the embedded flow.

For this thesis, we will only analyze the Redirect flow.

6.1 Authentication Protocol

This section will analyze Dokobit Identity Gateway, redirect-based user flow. The general overview of which can be seen in Figure 15. 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 6.

```
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",
  "session_token": "02
     f922c9917231ea8acbbbcf63796924af548c801d75772f2b1701b413462c61
     ",
```

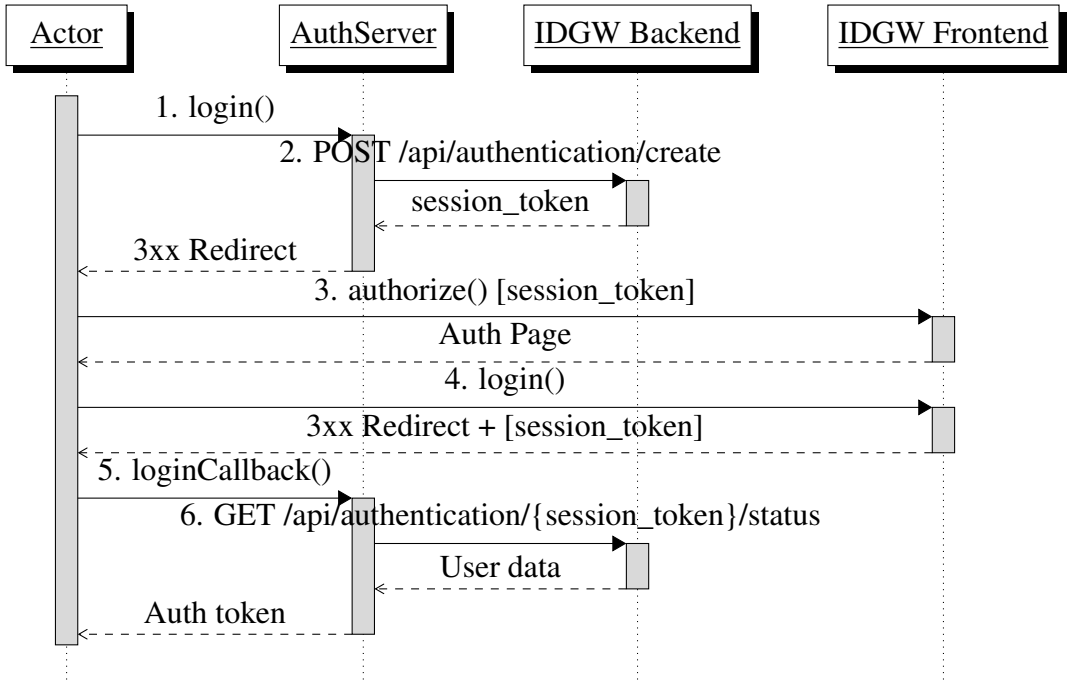


Figure 15. Dokobit Identity Gateway - Redirect-based user flow [71]

```
  "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 7.

```
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

6.2 Trust Anchor

Dokobit assumes the role of being the trust anchor. This role 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 it would be harder for adversaries to break into the system via external means.

Dokobit additionally provides the user's certificate from authentication, allowing companies to have local certificate validation to protect themselves if the identity provider gets compromised. This presence of the certificate makes this protocol potentially more secure than eeID.

6.3 Pricing

Dokobit is a commercial product, and therefore it has associated usage costs. In 2022, these costs are as seen in the table 1.

Table 1. 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 €.

6.4 Security Requirements

Of the three case studies analyzed in the thesis, Dokobit Identity Gateway was the only one that did not provide any request validation requirements to the relying party, only a single integration example [71]. This flow, however, is very similar to the one used by the eeID service and will be analyzed by using the same framework [4].

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.

6.4.1 Protocol's built-in security features

Replay attacks

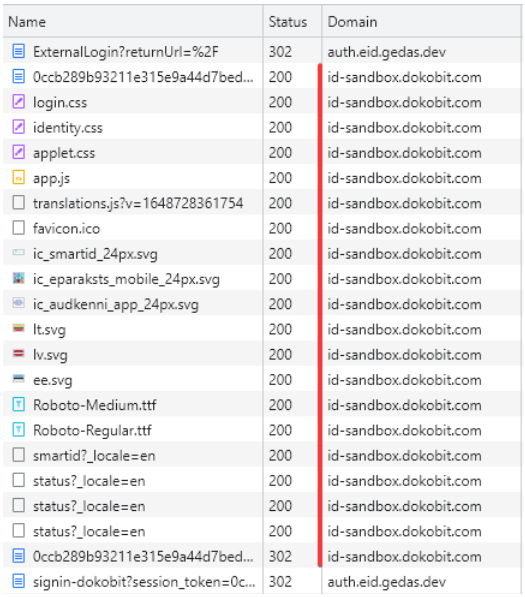
- Identity Gateway will reject the second GET request with the session id.

The developer does not have to implement state management to verify that a given session token was used only once, as Dokobit will reject subsequent requests. 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.

Attackers cannot influence the redirect URI. If companies use a static length redirect URI, the 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 16. 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 (all resources used in their service are from their domain, as seen in Figure 16). The developers of relying parties must also not embed third-party resources in the critical authentication pages.

Credential Leakage via Browser History

- Session ids are stored in browser history (see figure 17); 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 countermeasures.

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

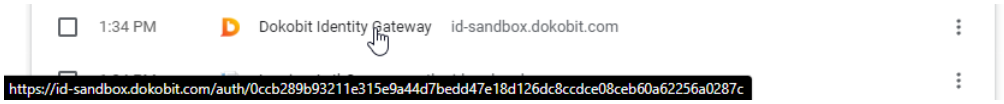


Figure 17. Session id is leaked inside the browser history when using Dokobit authentication

- The protocol protects from session token injection attacks by not allowing multiple requests to the authorize endpoint.
- It is impossible to steal a victim's or inject their session when the relying party has placed sufficient countermeasures.

The countermeasure against CSRF hinges on the fact that attackers do not know the final redirect URL. After the user agent returns to the callback URL, it may have a nonce value attached, generated during the initial request. If the query parameters match, the HTTP request is likely legitimate.

Dokobit provides **sufficient** tools to protect against CSRF attacks, and it is up to the relying party to integrate final mitigation measures.

Clickjacking

- Dokobit auth page does not use Content-Security-Policy. However, it does use header X-Frame-Option: SAMEORIGIN (see figure 18).
- The SAMEORIGIN feature is used in conjunction with CORS to support embedded flow; however, it would be better if X-Frame-Option was set to NONE.

Mitigation measures are sufficient on almost all browsers released in the last ten years [58]. The relying party should also place these headers on their log-in flow pages.

6.4.2 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 features if those validations do not match. This section will contain all of the features the RP needs to address.

Misconfiguration Attacks

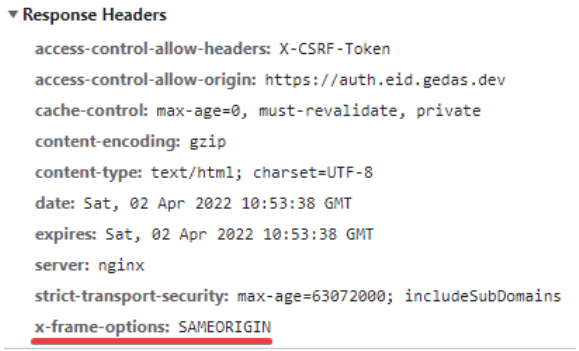


Figure 18. Response headers browsers receive when opening the eeID service's login page

Incorrect URL specified The relying party must 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 are 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 bind the Dokobit session_id to the user agent session.

Attack description For reference to how the mitigation works, we will use the figure 15. We will provide three examples for the countermeasure: without and with session binding, using a nonce.

Our attack exists right before the final redirect (right before step 5 in the figure 15). 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 only about tokens that have been authenticated but not consumed, which puts us right before step 5.

Attack #1 - No binding If the application uses static or a predictable redirect URI, attackers can forge it to contain a session token inside the query parameters. 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.

Attack #3 - With user agent binding and unpredictable URL When generating a redirect URL, the application creates a secure fixed-length random nonce and appends it to the query parameters of the return URL. For example, if the return URL were `https://my.app/callback`, now it would be `https://my.app/callback?nonce=0c723a`. This nonce will be bound to the user agent session.

With a predictable URL, attackers can still perform session token injection attacks. Suppose the attackers convinced a victim to use their session and subsequently blocked their access to WorkAuth servers after successful authentication. In that case, they could use the predictable redirect URL to finish the authentication process.

Including a nonce or any other kind of randomness in the authentication process requires attackers to guess the final URL, preventing them from consuming the session token given sufficiently high randomness.

6.5 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 8 and 9 are akin to data flow in figure 15. Numbers in comments directly correspond to the steps in the flow.

The first code block (listing 8) has three main parts:

1. establish a session with Dokobit as described in listing 6;
2. append an **encrypted** session cookie which would bind the browser to that specific Dokobit session (this cookie stores the encrypted session_token and nonce);
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 nonce = Convert.ToBase64String(RandomNumberGenerator.
            GetBytes(24)).TrimEnd('=').Replace('+', '-').Replace('/', '_
            ');
        var body = JsonSerializer.Serialize(new { return_url =
            QueryHelpers.AddQueryString(BuildRedirectUri(Options.
            CallbackPath), "nonce", nonce) });
        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 and nonce to encrypted cookie properties
        properties.Items["session_token"] = root.GetString("
            session_token");
        properties.Items["nonce"] = nonce;

        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 9). 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 and nonce received from cookie and query parameters match - this prevents attackers from injecting session tokens;

4. redeem the session_token and receive user data as described in listing 7;

```
// [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 and nonce received matches the one
        we initially saved
    var sessionToken = Request.Query["session_token"];
    if (properties.Items["session_token"] != sessionToken)
        return HandleRequestResult.Fail("Unexpected session_token
            received", properties);

    var nonce = Request.Query["nonce"];
    if (properties.Items["nonce"] != nonce)
        return HandleRequestResult.Fail("Unexpected nonce 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);

    var ticket = await CreateTicketAsync(new ClaimsIdentity(
        ClaimsIssuer), properties, user.RootElement);
    return HandleRequestResult.Success(ticket);
}
```

Listing 9. Handling access token creation

After completing these steps, the final code will be resilient to CSRF and session token injection attacks.

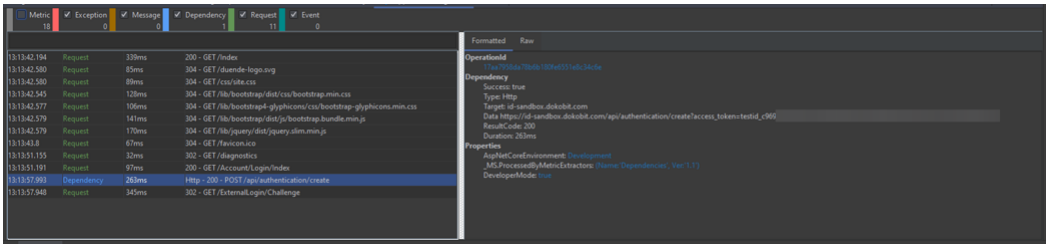


Figure 19. Example of data that would be sent to MS Azure Application Insights

6.6 Discovered weaknesses

6.6.1 API key transferred via query parameters

Dokobit Identity Gateway requires relying parties to send access tokens (API keys) via query parameters. When looking at the OAuth2.0 security best practices document [4], under section 4.3.2, we can see that it may lead to Credential Leakage via Browser History.

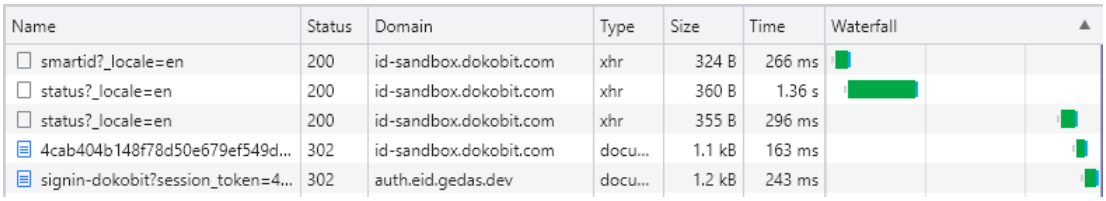
No browser is involved when using the API key. However, it does not mean no traces are left anywhere, though. Companies frequently use application and network logging solutions to detect outages. Some logging solutions may be a bit overzealous when deciding what they should be allowed to log.

We will use .NET 6 Web API and Application Insights to provide a concrete example. All dependency calls will be tracked and logged to Application Insights log storage on MS Azure with an out-of-the-box configuration. The data of dependency calls include the request path + query. The inclusion of query parameters would make the application leak the Dokobit API key by accident to Azure services (see figure 19). Improper access rules (allowing developers to troubleshoot issues in the application) would allow people with access to read the API key.

The impact of this attack appears to be insignificant, as it would only allow users to obtain the user's personal information, something an attacker could do with their own API key. The only other benefit would be sabotage by adding high usage costs to WorkAuth by rapidly forging requests and going around any rate limits the relying party may impose.

Disclosure

- Mar 02, 2022: Informed Dokobit CIRT about the exploit.
- Mar 13, 2022: Dokobit CIRT acknowledged the issue but decided against taking any action.



Name	Status	Domain	Type	Size	Time	Waterfall
smartid?_locale=en	200	id-sandbox.dokobit.com	xhr	324 B	266 ms	
status?_locale=en	200	id-sandbox.dokobit.com	xhr	360 B	1.36 s	
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 20. Waterfall of Dokobit Smart-ID authentication

6.6.2 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 vulnerability by performing these steps [72]:

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 [72].

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.

We need to look at the waterfall diagram of Identity Gateway's requests (see Figure 20) 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

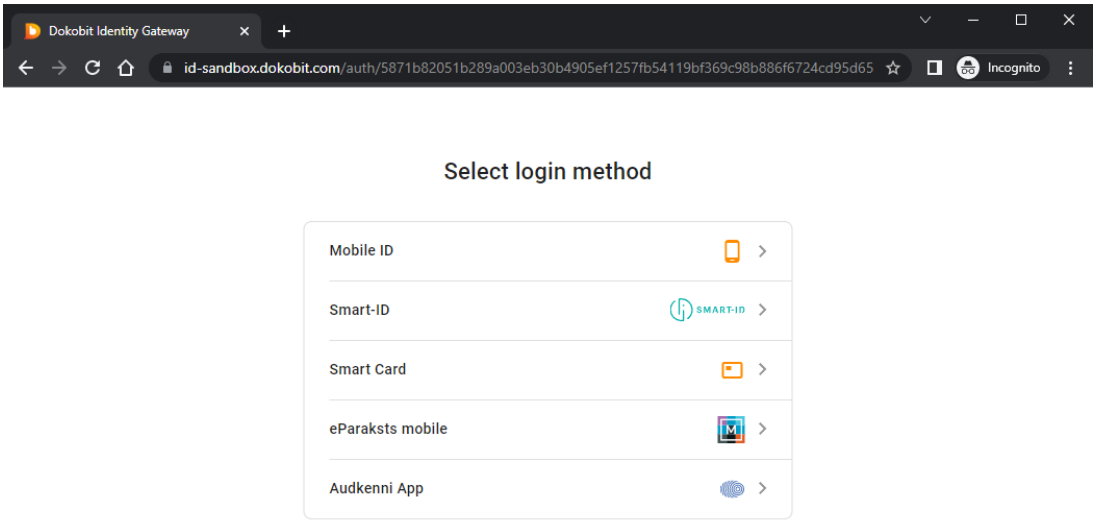


Figure 21. Authorization page as seen on the first visit

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

When a user refreshes the page, status checks are performed immediately, so if an attacker refreshed the page in the short one-second window, they would manage to 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.

The fix On Apr 11, 2022, Dokobit developers fixed the issue by allowing only one HTTP request to the authorize endpoint. This fix prevents attackers from abusing the automatic redirect feature, eliminating this exploit.

Figures 21 and 22 illustrate how the page looks after the fix was deployed.

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.
- Apr 11, 2022: Dokobit has fixed the issue.

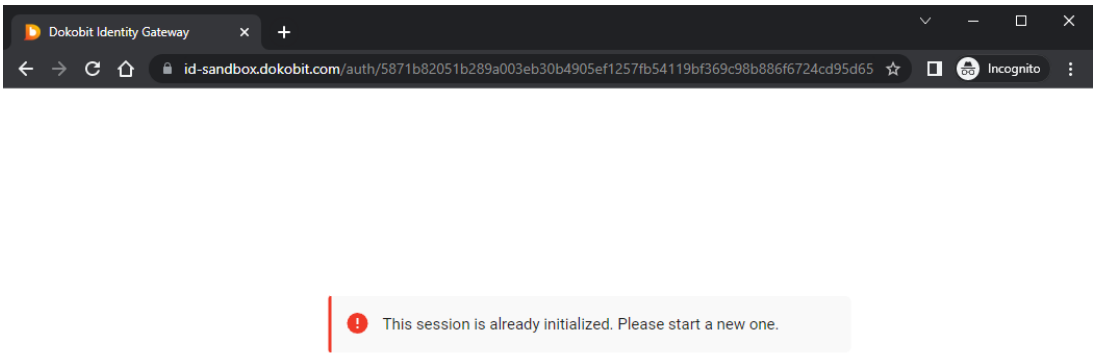


Figure 22. Authorization page as seen on subsequent visits

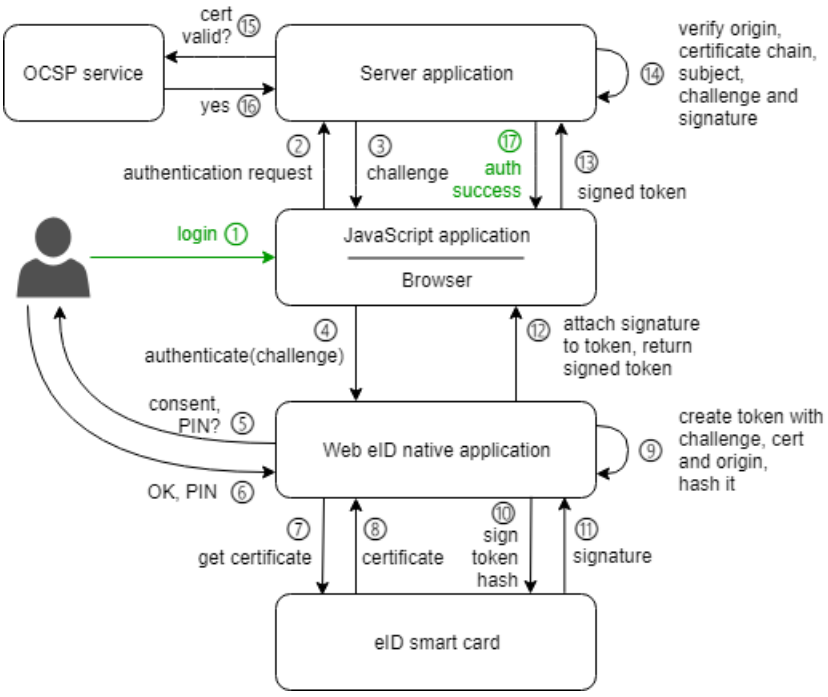


Figure 23. Web eID Authentication flow [73]

7 Case Study: Web eID

Released in the Summer of 2021 [22] and having undergone significant changes in January of 2022, this eID solution allows users to authenticate and sign documents using their country's smart cards.

Functionally this software solution 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 country's eID solutions. Supported countries include Estonia, Latvia, Lithuania, and Finland [22].

This service is built by the Estonian Information System Authority.

7.1 Authentication Protocol

Figure 23 displays the high-level overview of the complete flow of data within the Web eID system. A detailed explanation of the steps can be found in the technical specification [73]. Companies implementing the framework should only pay attention to the browser and the server application (steps 1-3 and 13-17).

7.2 Trust Anchor

Unlike Dokobit and eeID, Web eID does not provide any guarantees about the trustworthiness of a certificate. It requires the relying party to manually verify the received identity certificate [74]. The verification process involves checking the origin, certificate expiry, trust chain, OCSP response, and the signed challenge token.

One advantage of using Web eID is that there are no third-party intermediate services, which reduces the number of potential attack vectors, making it theoretically more secure than both eeID and Dokobit. The identity information comes directly from the device storing it. This benefit, however, comes with a comparatively massive implementation and maintenance cost, as the developers would have to create and maintain the trusted certificate store.

7.3 Pricing

The eID solutions enabled by Web eID software are free of charge to use, as the only external validation required, the OCSP [20] requests, are free.

7.4 Security Requirements

The data flow used by Web eID is fundamentally different from the used by the other two eID solutions. The main difference is that the server should not trust the identity information received from the eID provider. This difference makes using the IETF guidelines document [4] less valuable as it does not cover the case of what to do when the identity information is not trusted. Fortunately, RIA provides an exceptional integration and hardening guide [73], which we will use for our analysis.

Communication channel The Web eID framework is the only one covered by this thesis that uses an insecure communication channel. The reason why is it not secure is because the user or user agent can freely modify the identity data they send to the relying party. Developers must take caution and verify received data when integrating this eID solution. This channel is highly susceptible to man-in-the-middle, forgery, or other attacks done by a malicious end user.

7.4.1 Requirements for the identity provider

In Web eID's case, the identity provider is the Javascript library and the software used to bridge the connection between the library and the smart cards. Because the library can and should be embedded into the log-in page, the user is not required to leave the company's website.

Additionally, because the identity provider is on the same website as the relying party, it does not make sense to discuss the requirements separately, as was the case for eeID and Dokobit. For this reason, we will cover all common protocol attacks in the next section.

7.4.2 Requirements for the relying party

Web eID does not have a hosted website to verify security of. Instead, we will check if the integration documentation covers all common attacks.

Replay attacks

- The protocol requires the server to send a challenge nonce to the Web eID software.
- This generated nonce must be between 32 and 96 bytes (inclusive) in length [75].

The software allows for the use of nonces more than once. It is the server's responsibility to create nonces; make sure that they were not already used or expired.

The documentation states this explicitly: "Cache must be used for protection against replay attacks by guaranteeing that each authentication token can be used exactly once" [74].

Insufficient Redirect URI Validation

- When signing the challenge nonce, the Web eID library hashes and signs location.origin variable in addition to the nonce.

While not entirely redirect URI validation, the protocol still requires the domain check to mitigate against man-in-the-middle and authentication token export and replay attacks [73].

Cross-Site Request Forgery

- Documentation requires that "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" [74].

The document does not specify how companies should prevent CSRF, only that they must. However, the documentation does come with a comprehensive implementation example the developers can use as a reference.

Certificate injection Web eID, unlike the eID solutions analyzed before, comes with a hazardous problem. The client has complete control over the certificate they send and, if the certificate is not sufficiently validated, has the potential to impersonate anyone. Arnis Paršovs demonstrated an exploit of a similar variety back in 2015 [11]. Fortunately, the Web eID documentation provides validation steps the relying party must verify to mitigate this form of attack [73]:

1. the current time falls within the authentication certificate's validity period;
2. the purpose of the authentication certificate's key usage is client authentication;
3. the authentication certificate does not contain any disallowed policies;
4. the authentication certificate is signed by a trusted certificate authority;
5. certificate is not revoked (OCSP);
6. signed challenge nonce corresponds to the certificate's public key.

The documentation is thorough, listing all requirements developers must take to harden their systems. Additionally, documentation also provides reasons for the requirements' inclusion. Unfortunately, this list of validations is far longer than the other two eID solutions mentioned earlier, making it more challenging to integrate.

7.5 Integration

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

7.5.1 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 requirements the relying party must fulfil:

1. Generated challenge nonce must be between 32 and 96 bytes (inclusive) in length [75];
2. It must be guaranteed that the authentication token is received from the same browser to which the corresponding challenge nonce was issued [74]. The eID solution 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 [74].

In the implementation example, these measures were addressed by:

1. a 64 byte cryptographically secure randomly generated nonce is created (see listing 10);
2. challenge nonce is set in the user's session, which adversaries cannot access or 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 11);

```
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

7.5.2 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 [74];
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 our implementation, 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 [76]. A JavaScript application can only access the protected endpoints by providing RequestVerificationToken header (see listing 11);
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 section 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

7.5.3 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 [74]. Libraries can come with security vulnerabilities, and developers are reluctant to update their used version; however, it is still more favorable than to create vulnerabilities from misconfiguration [77].

The eu.webeid.security Java package performs most of the certificate validation: expiry, purpose, policy, OCSP [74]. 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 13).

```
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

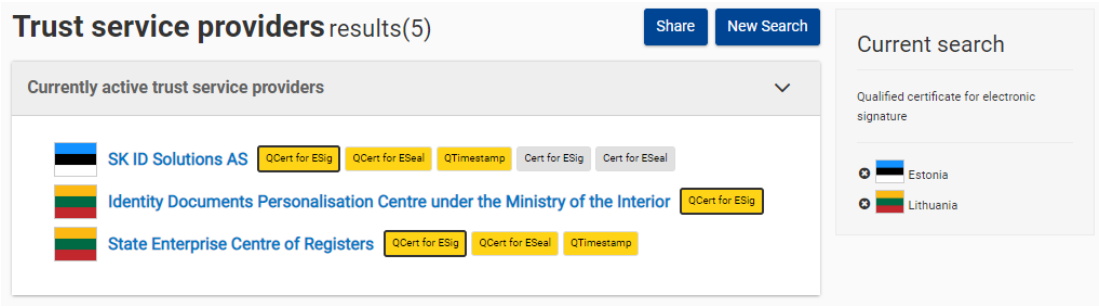


Figure 24. List of EU Trust service providers of Estonia and Lithuania capable of creating qualified certificates for e-signatures

The token validation service described in listing 13 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 [23]. 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 24). It is in stark contrast to Spain's 31 [23]. It is possible to further narrow down to only certificate generation services for qualified certificates (CA/QC); however, it would not be possible to narrow down anymore. In the case of Estonia's single TSP, we can see that only 3 CA are currently operational (see figure 25).

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 [78] who issues two certificates 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 [79]. 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 [80] or other issues.

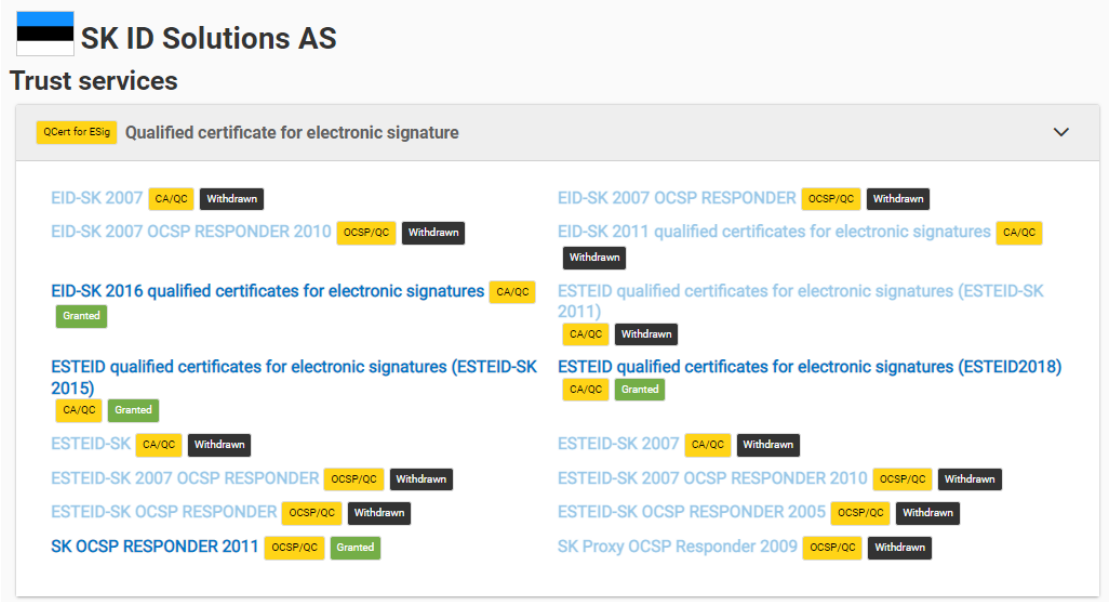


Figure 25. List of certificates issued to SK ID Solutions AS for the purposes of Qualified certificate for electronic signature

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. Zero trust architecture [81] is an excellent choice for this task.

	eeID	Dokobit	Web eID
Trust	government-backed third-party eID aggregation service	private third-party eID aggregation service	uses trust service providers directly
Market reach	multiple schemes; one country	multiple schemes; multiple countries	one common scheme; multiple countries
Price	(Most) Expensive	Expensive	Free
Attack resilience	High	High	High
Integration complexity	Easy	Easiest	Very difficult
Platform limitations	None	None	Requires special software

Table 2. Business features comparison of three eID solutions: eeID, Dokobit, and Web eID

8 Discussion

This section will summarize our findings, compare the different eID solutions and discuss the practical use cases for integration.

8.1 Comparison of the eID solutions

We have performed an in-depth integration analysis of three eID solutions - eeID, Dokobit, and Web eID. Each of these providers has some advantage over the competition.

8.1.1 Business features comparison

Table 2 displays a summary of each provider.

eeID The eeID service is an excellent option if the target audience is the residents of Estonia. It is a service created and maintained by one of Estonia's governmental institutions, supports all authentication schemes widely used in the target demographic, and is easy to integrate. The main downside of this service is its pricing - it is by far the most expensive service of the three.

Dokobit The eeID service is an excellent option if the target audience is the same as eeID or more international. It is a service created and maintained by a private company in Lithuania, supports the widest variety of authentication schemes and countries, and is the

	eeID	Dokobit	Web eID
Integration documentation clarity	Contradictory, no code examples	Clear, with examples	Clear, with examples
Hardening documentation clarity	Exists, does not explain why to do some steps	Does not exist	Clear, with explanations

Table 3. Documentation clarity comparison of three eID solutions: eeID, Dokobit, and Web eID

most straightforward service to integrate of the three. The biggest issue for companies to adopt this service would be to accept the risk of transferring sensitive data to a third-party private company.

Web eID The Web eID service is very different from the two others as it does not involve a third party, which significantly boosts security and trust. Another great advantage of this solution is that it is free to use. The main limitations of this eID solution are the comparatively weak market reach, the requirement for users to have software installed that they may not have yet, and the relatively astronomical integration complexity.

8.1.2 Developement features comparison

The previous section talks about business features. This section compares the information relevant to integration - documentation quality. Table 3 displays a short comparison of the three providers.

From table 3 we see that Web eID, although being the most complicated to integrate, provided by far the best documentation of all eID solutions.

In contrast, Dokobit, while providing great integration instructions, did not give any information on how to harden the security, leaving this the responsibility of the developers to figure it out on their own.

The integration of the eeID service has left me with more questions than answers. The documentation was thorough and did cover all of the attacks, although some parts were confusing. It felt like we were doing random tasks just because a book told me to. An explanation as to why some steps are required would massively benefit the readability of the documentation.

8.1.3 Summary

There is no best option for choosing an eID solution provider. Ultimately, the final decision comes on the priorities of the business.

If the highest degree of trust factor is required, a company will have no other option other than to use a QTSP, which in our case would be Web eID. Additionally, this option is likely to be suitable for those who wish to cut costs as much as possible, as for some QTSPs, the authentication operational costs are zero.

If the highest market reach is everything, adequately vetted third-party providers are the best choice. The most tangible difference between Dokobit and eeID is their market reach, and, by having a more extensive selection of supported schemes, Dokobit is the favorable option over eeID.

The only advantage eeID has over Dokobit is related to trust. It is impossible to measure trustworthiness, but if we consider that privately-held companies have more freedom over their operations over government institutions, the general public may perceive them as less trustworthy [68]. This perception could influence some companies to choose the latter option.

8.2 System architecture prerequisites

When designing the initial system, it quickly became apparent that eID is a very restricting form of authentication. To illustrate this, we have two examples.

Internal access Imagine that a company allows employees to access to read data on the database. The employees can access the database after the authentication process. Usually, this access is achieved by performing Active Directory authentication or simply using a username and password.

To adequately protect the data from confidentiality breaches, the eID authentication must be performed on the database level itself, a feature that would be exceptionally difficult to integrate. As an alternative, this company would restrict access to the databases and only allows users to interface with the databases via highly audited, custom-built solutions.

This solution now encounters a different political issue - is it justifiable to invest time and effort only to introduce arbitrary hurdles for employees to do their job? There are more straightforward solutions to having stricter access control and confidentiality breaches, both legally and technologically.

External access Imagine a company that lets users open storefronts. These users can theoretically sell anything, so the government would be keen to know the sellers' identities. To avoid getting into legal troubles with the government, the website owners are enticed to enable eID authentication for putting up sales listings. This way, the storefront hosting company would have very high confidence in the user's identity.

This situation suffers from the same issue as the internal access scenario. Because the eID authentication is not performed on the database level, and it would be impossible

to prove that it was, a rogue employee could frame someone by creating a listing for someone, as the access to the database is no longer restricted by eID.

With this limitation, eID authentication becomes useless without proof of data integrity. Digital signatures or a blockchain solution would potentially fix these cases. Further research on this idea is required.

8.3 Alternative to eID authentication

If we return to the original thesis idea of having one-time authentication, there might be an even better identity proof option for most companies.

The core issue with eID authentication In the thesis, we covered only half of the data access flow. Secure authentication methods are essential, but authorization is equally as important.

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 in the first place. 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 does register at the time they would expect. What should a 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, the company could assign employees to 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 essence, manual confirmation is unavoidable when setting up authorization rules.

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; they could be a privacy policy, terms of service, or an empty text file.

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 they already successfully verified their identity during enrollment.

This flow fully covers the initially proposed use case for eIDs, making them redundant. This approach is also 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 after the initial registration. However, companies can tackle this issue by hardening internal operational security.

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

9 Conclusion

This thesis aims to answer the question of what is the best eID authentication solution in Estonia available to the private sector.

To answer that question, we first compiled a list of all eID solutions available to Estonia and filtered out those who offer services to the private sector. This list included bank link, smart cards, Mobile-ID, Smart-ID, eeID, and Dokobit. For the in-depth analysis, we chose eeID, Dokobit, and smart cards in the form of Web eID.

We investigated the architectural requirements to support eID authentication and discovered that a simple SSO server is sufficient to integrate the authentication protocol fully. However, upon further research, we found that eID authentication does not cover data confidentiality or integrity requirements if the attacker is a malicious employee. They are likely to have alternative means to access the data store, allowing them to read and tamper with data without the use of eID.

We compared three eID solutions: eeID, Dokobit, and Web eID. The chosen comparison criteria were trust, market reach, price, convenience, integration and maintenance costs, security, and documentation comprehensiveness.

Web eID was the most trustworthy service and was the cheapest to operate. However, it suffered from comparatively high integration and maintenance costs alongside being most unfriendly to users and having the lowest market reach.

Dokobit was the cheaper paid option and had the highest market reach potential, spanning multiple countries. It was also the easiest to integrate of the three. The biggest issue with this service is the matter of trust - customers may not be willing to trust a private third-party company with their personal information.

The eeID service lies somewhere between the other two solutions. It is most similar to Dokobit, but it has a poorer market reach and is more expensive. The only advantage it has over Dokobit is that people may be more willing to give their information to a government institution instead of a private company.

The documentation analysis found that Web eID had the best and most understandable documentation, followed by eeID for unclear and contradictory parts. Lastly, Dokobit did the worst by not providing any information about the hardening process.

All of the analyzed services passed the security assessment at the end of the research. Web eID and eeID did not have any issues. Dokobit had a CSRF vulnerability on their authorization endpoint, which we disclosed and they fixed.

Summary The ability to integrate eID in the private sector exists; however, the public acceptance of them is limited. We were unable to find any technological limitations stopping widespread adoption. Poor market adoption is likely caused by political, economic, social, or legal reasons.

References

[1] Adam Ozimek. The future of remote work. *Available at SSRN 3638597*, 2020.

[2] British Council. IELTS - how to register. https://www.ielts.org/for-test-takers/how-to-register, 2022. Online; accessed 26-Feb-2022.

[3] THE EUROPEAN PARLIAMENT and THE COUNCIL OF THE EUROPEAN UNION. Regulation (EU) No 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC. *Official Journal of the European Union*, L 257/73, 2014.

[4] Torsten Lodderstedt, John Bradley, Andrey Labunets, and Daniel Fett. OAuth 2.0 Security Best Current Practice. Internet-Draft draft-ietf-oauth-security-topics-19, Internet Engineering Task Force, December 2021. Work in Progress.

[5] Dick Hardt. The OAuth 2.0 Authorization Framework. RFC 6749, October 2012.

[6] Riigikogu. Identity documents act. 2021.

[7] European Comission. Digital economy and society index (desi) 2021. 2021.

[8] Răzvan Dan. Cât plătesc oamenii pentru buletinul cu cip care a început să fie eliberat deja populației. https://stirileprotv.ro/stiri/actualitate/buletinul-cu-cip-eliberat-deja-in-cluj-napoca-oamenii-platesc-70-de-lei-pentru-
html, 2021. Online; accessed 27-Feb-2022.

[9] Informacinės visuomenės plėtros komitetas. E-government gateway. https://www.epaslaugos.lt/portal/nlogin, 2022. Online; accessed 27-Feb-2022.

[10] Estonian Information System Authority. As of 1 march, it will no longer be possible to access certain public e-services via a bank link. https://www.ria.ee/en/news/1-march-it-will-no-longer-be-possible-access-certain-public-e-services-bank-link
html, 2021. Online; accessed 27-Feb-2022.

[11] Arnis Parsovs. SEB Estonia Internet bank ID card authentication bypass. https://youtu.be/rRB8jZnS5nY, 2015. Online; accessed 30-Mar-2022.

[12] Jesus Carretero, Guillermo Izquierdo-Moreno, Mario Vasile-Cabezas, and Javier Garcia-Blas. Federated identity architecture of the european eid system. *IEEE Access*, 6:75302–75326, 2018.

[13] European Comission. Electronic identification schemes notified pursuant to Article 9(1) of Regulation (EU) No 910/2014 of the European Parliament and of the Council on electronic identification and trust services for electronic transactions in the internal market. *Official Journal of the European Union*, 2022/C 78 I/02, 2022.

[14] Applied CyberSecurity Group. Estonian e-services using eid. https://acs.cs.ut.ee/services_using_eid, 2021. Online; accessed 21-Nov-2021.

[15] Katri Kerem. *Internet banking in Estonia*, volume 7. PRAXIS, 2003.

[16] AS SEB Pank. Bank link and authentication service. https://www.seb.ee/en/bank-link, 2021. Online; accessed 21-Nov-2021.

[17] Arnis Paršsovs. Security analysis of internet bank authentication protocols and their implementations. Master's thesis, Tallinn University of Technology, 2012.

[18] Estonian Information System Authority. Home - id.ee. https://www.id.ee/en/, 2021. Online; accessed 20-Nov-2021.

[19] Statistics Estonia. Population figure. https://www.stat.ee/en/avasta-statistikat/valdkonnad/rahvastik/population-figure, 2021. Online; accessed 20-Nov-2021.

[20] Stefan Santesson, Michael Myers, Rich Ankney, Ambarish Malpani, Slava Galperin, and Dr. Carlisle Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. RFC 6960, June 2013.

[21] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, August 2018.

[22] Estonian Information System Authority. Web eid. https://www.id.ee/en/article/web-eid/, 2022. Online; accessed 28-Feb-2022.

[23] European Commission. Trust service providers. https://esignature.ec.europa.eu/efda/tl-browser/, 2022. Online; accessed 28-Feb-2022.

[24] SK ID Solutions. History - 2007. https://www.skidsolutions.eu/en/about/history/year-2007/, 2007. Online; accessed 21-Nov-2021.

[25] SK ID Solutions. Price list of Mobile-ID service. https://www.skidsolutions.eu/en/services/pricelist/mobile-id-service, 2021. Online; accessed 21-Nov-2021.

[26] Telia. Mobiil-ID. https://www.telia.ee/era/lisateenused/mobiil-id/, 2022. Online; accessed 28-Feb-2022.

[27] SK ID Solutions. History - 2017. https://www.skidsolutions.eu/en/about/history/year-2017/, 2017. Online; accessed 21-Nov-2021.

[28] SK ID Solutions. Smart-ID documentation. https://github.com/SK-EID/smart-id-documentation, 2021. Online; accessed 10-Feb-2022.

[29] Sylvie Lacroix, Olivier Delos, Evgenia Nikolouzou (ENISA), and Slawomir Gorniak (ENISA). Assessment of Standards related to eIDAS. Technical report, European Union Agency For Network and Information Security, 2018.

[30] SK ID Solutions. Digital signatures with Smart-ID. https://www.smart-id.com/e-service-providers/smart-id-as-a-qscd/, 2018. Online; accessed 28-Feb-2022.

[31] SK ID Solutions. Price list of Smart-ID service. https://www.skidsolutions.eu/en/services/pricelist/smart-id/, 2021. Online; accessed 21-Nov-2021.

[32] Estonian Information System Authority. The information system authority's authentication service TARA. https://www.ria.ee/en/state-information-system/eid/partners.html, 2022. Online; accessed 27-Feb-2022.

[33] Estonian Information System Authority. TARA business description. https://e-gov.github.io/TARA-Doku/Arikirjeldus, 2021. Online; accessed 21-Nov-2021.

[34] Estonian Information System Authority. TARA technical specification. https://e-gov.github.io/TARA-Doku/TechnicalSpecification, 2021. Online; accessed 21-Nov-2021.

[35] Natsuhiko Sakimura, John Bradley, Mike Jones, Breno De Medeiros, and Chuck Mortimore. Openid connect core 1.0. *The OpenID Foundation*, 2014.

[36] Estonian Internet Foundation. eeid service. https://www.internet.ee/eeid-service, 2021. Online; accessed 28-Feb-2022.

[37] Estonian Internet Foundation. Subscription agreement and terms of use of the estonian internet foundation electronic identification service. https://meedia.internet.ee/files/Pricing_eeID.pdf, 2021. Online; accessed 28-Feb-2022.

[38] Estonian Ministry of Education and Research. HarID. https://harid.ee/et/users/sign_in, 2022. Online; accessed 28-Feb-2022.

[39] Dokobit. Electronic signature API solutions. https://www.dokobit.com/solutions, 2022. Online; accessed 28-Feb-2022.

[40] Dokobit. Dokobit authentication solutions pricing. https://www.dokobit.com/docs/pricing/Dokobit-authentication-solutions-pricing.pdf, 2022. Online; accessed 28-Feb-2022.

[41] Information technology — Security techniques — Entity authentication assurance framework. Standard, International Organization for Standardization, Geneva, CH, 2013.

[42] European Comission. COMMISSION IMPLEMENTING REGULATION (EU) 2015/1502 of 8 September 2015 on setting out minimum technical specifications and procedures for assurance levels for electronic identification means pursuant to Article 8(3) of Regulation (EU) No 910/2014 of the European Parliament and of the Council on electronic identification and trust services for electronic transactions in the internal market. *Official Journal of the European Union*, L 235/7, 2015.

[43] THE EUROPEAN PARLIAMENT and THE COUNCIL OF THE EUROPEAN UNION. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC. *Official Journal of the European Union*, L 119/1, 2016.

[44] Bernd Zwattendorfer and Daniel Slamanig. The austrian eid ecosystem in the public cloud: How to obtain privacy while preserving practicality. *Journal of Information Security and Applications*, 27-28:35–53, 2016. Special Issues on Security and Privacy in Cloud Computing.

[45] Herbert Leitold. eid and egovernment in austria. Graz University of Technology https://online.tugraz.at/tug_online/voe_main2.getvolltext?pCurrPk=16600, 2006. Online; accessed 01-Mar-2022.

[46] Martín J.C., Aranda N.I., Carreras R.C., Pasic A., Baún J.C.P., Ksystra K., Triantafyllou N., Papadakis H., Torroglosa E., and Ortiz J. *LEPS - Leveraging eID in the private sector*. River Publishers, 2019.

[47] Vicente Guerola-Navarro, Raúl Francisco Oltra Badenes, Hermenegildo Gil Gómez, and Doina Stratu-Strelet. Eid@ cloud: integration of electronic identification in european platforms in the cloud in accordance with the eidas regulation. *3C Business, Researchóand critical thinking*, 8(3):64–87, 2019.

[48] Jesus Carretero, Guillermo Izquierdo-Moreno, Mario Vasile-Cabezas, and Javier Garcia-Blas. Federated identity architecture of the european eid system. *IEEE Access*, 6:75302–75326, 2018.

[49] L. O'Gorman. Comparing passwords, tokens, and biometrics for user authentication. *Proceedings of the IEEE*, 91(12):2021–2040, 2003.

[50] Yuan Cao and Lin Yang. A survey of identity management technology. In *2010 IEEE International Conference on Information Theory and Information Security*, pages 287–293, 2010.

[51] Scott Cantor, Internet2 John Kemp, Nokia Rob Philpott, and E Maler. Assertions and protocols for the oasis security assertion markup language. *OASIS Standard (March 2005)*, pages 1–86, 2005.

[52] eIDAS eID Technical Subgroup. eidas saml attribute profile v1.2. Technical report, European Commission, 2019.

[53] Estonian Information System Authority. Data protection terms of the web eid website. https://web-eid.eu/files/Web%20eID%20privacy%20policy.pdf, 2021. Online; accessed 09-Mar-2022.

[54] Dokobit. Privacy policy. https://www.dokobit.com/compliance/privacy-policy, 2022. Online; accessed 09-Mar-2022.

[55] Adam Chester. Sql server authentication with metasploit and mitm. https://blog.xpnsec.com/sql-server-auth-mitm/, 2017. Online; accessed 09-Mar-2022.

[56] Microsoft Docs contributors. Enable encrypted connections to the database engine. https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/enable-encrypted-connections-to-the-database-engine, 2021. Online; accessed 09-Mar-2022.

[57] Gediminas Milasius. at_hash is not OIDC compliant. https://github.com/e-gov/TARA-Doku/pull/16, 2022. Online; accessed 01-Apr-2022.

[58] Alexis Deveria. Can I use X-Frame-Options HTTP header. https://caniuse.com/x-frame-options, 2022. Online; accessed 02-Apr-2022.

[59] OpenID Foundation. Certified OpenID Connect Implementations. https://openid.net/developers/certified/, 2022. Online; accessed 27-Mar-2022.

[60] ASP.NET Core Contributors. Microsoft.AspNetCore.Authentication.OpenIdConnect source code. https://github.com/dotnet/aspnetcore/tree/main/src/Security/Authentication/OpenIdConnect/src, 2022. Online; accessed 27-Mar-2022.

[61] OKTA Developer Documentation Contributors. OpenID Connect & OAuth 2.0 API. https://developer.okta.com/docs/reference/api/oidc/, 2022. Online; accessed 27-Mar-2022.

[62] Estonian Information System Authority. TARA-Client. https://github.com/e-gov/TARA-Client/blob/279cdce40aa470498b8db5b476ed99ec99147875/src/main/java/ee/ria/tara/mid/controller/DemoRestController.java#L177, 2022. Online; accessed 26-Mar-2022.

[63] Michael Jones, John Bradley, and Nat Sakimura. JSON Web Token (JWT). RFC 7519, May 2015.

[64] Arnis Parsovs. Security Analysis of RIA's Authentication Service TARA. Technical report, University of Tartu, 2021.

[65] European Union Intellectual Property Office. 017858226 - dokobit. https://euipo.europa.eu/eSearch/#details/trademarks/017858226, 2018. Online; accessed 20-Mar-2022.

[66] Dokobit. Dokobit - about us. https://www.dokobit.com/company/about-us, 2022. Online; accessed 20-Mar-2022.

[67] Bureau Veritas. Dokobit - iso/iec 27001:2013 certification. https://www.dokobit.com/docs/compliance/Dokobit-iso27001-certificate.pdf, 2019. Online; accessed 20-Mar-2022.

[68] Hong-Youl Ha. Factors influencing consumer perceptions of brand trust online. *Journal of product & brand management*, 2004.

[69] Microsoft Docs contributors. Centralized universal login vs. embedded login. https://auth0.com/docs/authenticate/login/universal-vs-embedded-login. Online; accessed 20-Mar-2022.

[70] W. Denniss and J. Bradley. OAuth 2.0 for Native Apps. RFC RFC8252, October 2017.

[71] Dokobit. Dokobit identity gateway api documentation. https://id-sandbox.dokobit.com/api/doc. Online; accessed 20-Mar-2022.

[72] Gediminas Milasius. Vulnerability in dokobit identity gateway - low tech. https://youtu.be/dG42UaHUoOE, 2022. Online; accessed 21-Mar-2022.

[73] Estonian Information System Authority. Web eid: electronic identity cards on the web. https://github.com/web-eid/web-eid-system-architecture-doc, 2022. Online; accessed 05-Mar-2022.

[74] Estonian Information System Authority. web-eid-app - authenticate.cpp. https://github.com/web-eid/web-eid-authtoken-validation-java, 2022. Online; accessed 05-Mar-2022.

[75] Estonian Information System Authority. web-eid-app. https://github.com/web-eid/web-eid-app/blob/8bedc0c57698c729012a0c16f590ce0907d27d20/src/controller/command-handlers/authenticate.cpp#L103, 2021. Online; accessed 05-Mar-2022.

[76] Fiyaz Hasan, Rick Anderson, and Steve Smith. Prevent cross-site request forgery (xsrf/csrf) attacks in asp.net core. https://docs.microsoft.com/en-us/aspnet/core/security/anti-request-forgery, 2022. Online; accessed 05-Mar-2022.

[77] Ying Wang, Bihuan Chen, Kaifeng Huang, Bowen Shi, Congying Xu, Xin Peng, Yijian Wu, and Yang Liu. An empirical study of usages, updates and risks of third-party libraries in java projects. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 35–45, 2020.

[78] Asmens dokumentų išrašymo centras. Asmens tapatybės kortelė ir elektroninis parašas. https://www.nsc.vrm.lt/downloads.htm, 2021. Online; accessed 05-Mar-2022.

[79] SK ID Solutions. Certificates. https://www.skidsolutions.eu/en/repository/certs/, 2022. Online; accessed 05-Mar-2022.

[80] Estonian Information System Authority. Roca vulnerability and eid: Lessons learned. Technical report, Estonian Information System Authority, 2018.

[81] Alper Kerman, Oliver Borchert, Scott Rose, and Allen Tan. Implementing a zero trust architecture. *National Institute of Standards and Technology (NIST)*, 2020.

Appendix

I. Glossary

- **CA**. Certificate Authority. A certificate capable of issuing other certificates the relying parties trust.
- **CSRF**. Cross-Site Request Forgery. A malicious request which executes an action on behalf of the victim.
- **eID**. Government-issued Electronic identity.
- **eIDAS**. Regulation (EU) 910/2014.
- **GDPR**. Regulation (EU) 2016/679.
- **Horizontal scaling**. Expanding a system not by getting better hardware but by getting more physical machines.
- **HTTP**. HyperText Transfer Protocol. A common protocol used to transfer information online.
- **IETF**. Internet Engineering Task Force. Standards organization.
- **IT**. Information Technology.
- **LoA**. Level of Assurance. A measure of how certain a party is about something.
- **MFA**. Multi-factor authentication. Additional security measure users must pass before they are granted access.
- **OCSP**. Online Certificate Status Protocol. Used for obtaining the revocation status of an X. 509 digital certificate.
- **PII**. Personally identifying information.
- **QES**. Qualifier Electronic Signature. A signature backed by a QTSP. Legally binding signatures in the EU.
- **QSCD**. Qualifier Electronic Signature Creation Device. A tool capable of creating QES.
- **QTS**. Qualified Trust Service. A trust anchor as defined in eIDAS. Acts as a final authority for determining if a property, such as a digital signature, is authentic.

- **QTSP**. Qualified Trust Service Provider. A natural or legal person providing one or more QTS.
- **RP**. Relying Party. An entity (such as a company) that relies on a trust anchor.
- **SSO**. Single sign-on. A centralized authentication solution issuing access tokens for multiple services.
- **TLS**. Transport Layer Security. Encryption mechanism used by HTTP and other protocols.
- **User Agent**. An application that performs actions on behalf of a user. In the context of Web browsing, it usually is a Web Browser.
- **WorkAuth**. A fictitious company used in the thesis wishing to integrate the eID solutions. Used to clear up ambiguity in some places.

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

I. Interview

This interview's goal is to understand better the reasons for the poor adoption of eIDs in the private sector.

We conducted this interview 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?

   Ashot: yes, fully aware. But you have to keep hardware always with you, besides that it can be lost or stolen. That's why MFA is a preferred way for authentications and it get global adoption.

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?

   Ashot: yes, all of them. Using Mobile-ID and Smart-ID all the time, Smart-ID is somehow more user-friendly. ID card - exceptional cases.

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?

   Ashot: don't forget that eID is a workable solution in Baltics, but most of the European countries are not so much advanced. In Switzerland I have to print every document and sign it offline. Even the TAX declaration.. this is nightmare

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?

   Ashot: absolutely yes

5. Authentication and signing usually come hand in hand, but if you were to have the ability to choose, assuming authentication and signing both cost 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?

   Ashot: investment in this case would make sense into signature

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?

   Ashot: absolutely yes - digital signature

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?

   Ashot: human mistakes can occur in both manual and digital scenarios. To avoid such issues the automated process can help to propose for signature only valid documents. If this process still fails, then second pair of eyes could be a solution. But in all cases it should be digital signature as a part of digital process

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?

   Ashot: acceptable, but I will look for additional solutions to have better coverage

   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?

   Ashot: it should be highly trusted middleware, but yes this is acceptable. Such solutions already exist for payments for example

   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?

Ashot: I would search for other solution with better coverage

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?

Ashot: no concerns if they can gain trust and also would be great to see support on the government/official level for such provider

Ashot: I would chose the second option as it can bring mass adoptions. But should be supported by government/officials

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?

   Ashot: users are already used to have such services close to "free of charge". how much is the owner of the business ready to pay for this? Not a lot. 10k authentications per month is around 300-400 active users. So this adds additional costs more than 1 EUR per user. Here would make sense S/M/L/Enterprise packages with different price tags

10. Options 2-4 also create digital signatures; the first cannot. Does your opinion change at all about which solution you would pick?

    Ashot: I stay with the largest coverage

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?

    Ashot: in case of internal IT solution - I could use internal ID system as I can verify all accounts. In case of public solution with a global coverage - you need something more official. We have an example of our partner - `https://www.farmerconnect.com/products` who is trying to introduce Farmer ID, but you need local presence and strict verification rules. This concept is close to failure

After we received the initial answers, we asked more questions based on the feedback:

1. Signatures » Authentication. Between the two, authentication is just not as useful, as it helps with confidentiality, whereas signing has legal status. If a solution does not offer signing functionality you would not even consider it.

   Ashot: we are using B2C today for auth, right? So one can exist without another. But I guess in scope of this project you signature is a must.

2. Trust. The solution must be supported by government/officials. Is ISO/IEC 20001:2013 certification sufficient? `https://www.dokobit.com/docs/compliance/Dokobit-iso27001-certificate.pdf`

   Ashot:not really. Such certificate is a minimum requirement. I would like to see that government portals, or banks are adopting this solution. This provides sufficient trust into the solution. Also you cannot just sign the document with the homemade tool. It should be legal in the country so you have to deal with local authorities. Otherwise your signature does not worth a penny.

3. Scope. The solution should have a large market reach, so you would focus on 3rd party service providers, rather than the primary trust sources like Smart-ID (SK ID Solutions)

   Ashot: Either there should be a global standard and solutions will support interoperability (e.g. I would be able to use smart-ID in Switzerland), or there should be an independent service provider which will get support from local authorities. Like the middleware solution for credit cards (I don't remember the name)

4. Pricing. It does have a significant impact, but it is not as important as first 3. Say if there was an option to skip authentication - use your own solution, and use the eID service provider for signatures only. The cost of a digital signature by using them costs around 30ct per signature. This would be around 1500 signatures per month at 500 eur. Is this a more appealing offer?

   Ashot: who is the target group? Is it a bank to offer this for own customers? Then it is too expensive for them I would say. Is it a business owner? Today they open PDF and apply PNG of signature into the file free of charge. Would they be ready to pay 500 for 1500 signatures - maybe, but most probably they will try to cut some costs here. Is it government? Then they have huge volumes, so it is too expensive (but they can use this an argument for green-environment)

5. Assuming we have signatures as 100% required feature, what impact, in your opinion, does the Authentication support/Trust/Scope/Pricing have in relation to one another? For example it can be 10%/50%/30%/10%. Maybe there are additional deal breakers?

Ashot: You cannot have signature without passing the authentication, right? So it is included in the package by default. So between Trust/Scope/Pricing I would say 40/30/30. In this case Trust also means that it is legally accepted and if I go to court - they will accept this signature. Also important easy-to use and friendliness, and plug-n-play approach. It should be also available on all possible devices.

II. 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, Estonian 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;

V. Source code

The full source code used in this thesis can be found at `https://github.com/GedasFX/eID-at-UT`. The source code for Web eID OCSP validation is in a different repository: `https://github.com/GedasFX/web-eid-token-validation-service`.

The website running this source code is at `https://auth.eid.gedas.dev/`.