UNIVERSITY OF TARTU
Institute of Computer Science
Cybersecurity Curriculum

**Elizabete Liene Šterna**

# Security Architecture of the Latvian eParaksts mobile

**Master's Thesis (24 ECTS)**

Supervisor: Arnis Paršovs, PhD

Tartu 2022

## Security Architecture of the Latvian eParaksts mobile

**Abstract:**

The *eParaksts mobile* is a Latvian eID solution that is used for authentication and electronic signature creation with more than 187 000 users. It can be used to access government e-services in Latvia and create qualified electronic signatures with the same legal strength as handwritten signatures. Since *eParaksts mobile* is not an open-source solution, there is no publically available information describing the architecture of *eParaksts mobile*.

Therefore, in this thesis, network traffic analysis is performed to understand and describe how the authentication and electronic signature creation schemes are implemented. This analysis depicts in detail the enrollment, authentication and electronic signature creation processes and shows that *eParaksts mobile* has a hybrid architecture – partly device-based, partly server-based. The private key for the authentication scheme is kept on the user's device, while the private key for signature creation is kept on an HSM on the server-side.

Additionally, a discussion of security implications emerging from the architecture of *eParaksts mobile* is provided. Moreover, this thesis provides a foundation for future studies of security analysis of the *eParaksts mobile* solution.

**Keywords:**

Authentication, cloud-based digital signature, electronic signature, eParaksts mobile, mobile eID, remote QSCD, trusted execution environment (TEE).

**CERCS:**

P170 Computer science, numerical analysis, systems, control

# Läti eParaksts mobile turvaarhitektuur

**Lühikokkuvõte:**

Läti *eParaksts mobile* on eID lahendus, mida kasutatakse autentimiseks ja elektrooniliste allkirjade loomiseks enam kui 187 000 kasutajaga. Seda saab kasutada Läti valitsuse e-teenustele juurdepääsuks ja kvalifitseeritud elektrooniliste allkirjade loomiseks, millel on samasugune juriidiline õigus kui käsitsi kirjutatud allkirjal. Kuna *eParaksts mobile* ei ole avatud lähtekoodiga lahendus, siis puudub ka avalikult kättesaadav *eParaksts mobile* arhitektuuri kirjeldav info.

Seetõttu tehakse käesolevas lõputöös võrguliikluse analüüs, et mõista ja kirjeldada, kuidas autentimis- ja elektroonilise allkirja loomise skeeme realiseeritakse. See analüüs kirjeldab üksikasjalikult registreerimise, autentimise ja elektroonilise allkirja loomise protsesse ning näitab, et eParaksts mobile on hübriid arhitektuuriga – osaliselt seadme-, osaliselt serveri põhiselt. Autentimisskeemi privaatvõtit hoitakse kasutaja seadmes, allkirja loomise privaatvõtit aga serveripoolses HSM-is.

Lisaks käsitletakse *eParaksts mobile* arhitektuurist tulenevaid turva mõjusid. Lisaks annab see lõputöö aluse *eParaksts mobile* lahenduse turvaanalüüsi tulevastele uuringutele.

**Võtmesõnad:**

Autentimine, pilvepõhine digiallkiri, elektrooniline allkiri, *eParaksts mobile*, mobiilne eID, kaug-QSCD, usaldusväärne täitmiskeskkond (TEE)

**CERCS:**

P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

## Table of Contents

# 1 Introduction

*eParaksts mobile* is a Latvian mobile electronic identity (eID) solution that is used for authentication and electronic signature creation. This solution consists of a mobile app for *Android* and *iOS* mobile platforms and the *eParaksts* server backend. Electronic signatures in Latvia were introduced in 2006 [1], and *eParaksts mobile* was launched in 2018 [2]. However, only from June 28th 2020, signatures created using *eParaksts mobile* have been recognised as qualified electronic signatures (QES) and are equivalent to handwritten signatures in the EU; until then, signatures made with *eParaksts mobile* were equivalent to handwritten signatures only in Latvia, but in the EU, such signatures had a status of advanced electronic signature with qualified certificate (AdES/QC) [3].

*eParaksts mobile* has been around for four years, and during this time, more than 187 000 people have applied for this tool, and around 10 000 people receive *eParaksts mobile* every month [4]. The popularity of *eParaksts mobile* is growing due to its convenience over traditional eID cards, especially in the last years amid the COVID-19 pandemic due to increased usage of online services. Despite its popularity, there has not been any academic research done regarding the security of *eParaksts mobile* or other currently used products of *eParaksts*.

The motivation of this study is as follows. This thesis contributes to Latvia's cyberspace since no academic research exists about *eParaksts mobile* in the literature. Additionally, such a study fills the research gap between different mobile eID solutions in Europe since only a handful of mobile eID solutions have been analysed in the literature. Moreover, considering the high heterogeneity in the current mobile eID landscape, it is challenging to identify trends and best practices for those in charge of the development and deployment of mobile eID solutions, hence this study can serve as a good example to other countries that wish to implement a mobile eID solution.

## 1.1 Problem statement

There are three research questions to be answered in this thesis:

1. How does the user authentication scheme work in *eParaksts mobile*?
2. How does the electronic signing scheme work in *eParaksts mobile*?
3. What is the architecture of the mobile eID solution *eParaksts mobile*?

The goal of the study is to understand and describe the user authentication and signing schemes in *eParaksts mobile*, specifically, to understand the architecture of the mobile eID solution – whether it is client-based or server-based, and what kind of requests are sent between the client (the *eParaksts mobile* app and the website) and the server, and what kind of information these requests contain. Currently, such information is not a part of publicly available documentation about the *eParaksts mobile* solution.

For the purpose of understanding the implementation of authentication and electronic signature creation schemes, the *eParaksts mobile* app for the *Android* platform will be analysed. The mobile apps for *Android* and *iOS* platforms likely share the same API since the solution is independent of the mobile platform used by the user. Analysis of the app for the *Android* platform will allow understanding of the overall architecture of the *eParaksts mobile* solution regardless of the mobile platform.

The study is limited to the analysis of authentication and signing protocols implemented in *eParaksts mobile,* as the goal of the study is not to analyse the entire mobile eID solution and application as such.

## 1.2 Outline

The thesis is organised as follows. Chapter 2 introduces different types of eID solutions and discusses other studies related to them. The third chapter describes the *eParaksts mobile* solution from the user's perspective to demonstrate the functionality of the solution. Next, Chapter 4 is dedicated to the necessary modifications made to the *eParaksts mobile* app and the experiment set up for the network traffic inspection. Chapter 5 describes the analysis of the network traffic of *eParaksts mobile* that is done to understand the implementation of the authentication and electronic signing schemes. In Chapter 6, the observed architecture of *eParaksts mobile* and authentication and electronic signing schemes are described. Chapter 7 focuses on relevant legal aspects of *eParaksts mobile* related to the eIDAS regulation. Finally, in Chapter 8, a discussion about security-related aspects of *eParaksts mobile* is provided, and Chapter 9 concludes the thesis.

## 2 Related work

Traditionally, eID schemes are based on Public Key Cryptography (PKI), and traditionally users' private keys are kept on a smart card [5]. Mobile eIDs as an alternative to smart-card based eID solutions became relevant since governments realised the inconvenience of smart-cards due to specific hardware requirements, for instance, card readers and special software [6], and with the advancements in technologies, mobile alternatives were explored. Based on where the private key that serves as the security token is located, mobile eID architectures can be classified as server-based, SIM-based or device-based [7, 8].

In the server-based architecture, a security token is implemented via a hardware security module (HSM) on the server-side which handles cryptographic operations. For server-based solutions, it is important how users are authenticated to the mobile eID solution because the strength of the mobile eID solution relies on how easily the security token can be accessed. Therefore, several solutions have been proposed in the literature investigating advantages and disadvantages as well as security assumptions based on used protocols and encryption schemes [9-12]. Austria is one of the European countries with a server-based mobile eID solution. In their first mobile eID solution, two-factor authentication was implemented – users had to provide their PIN and authorisation code that was sent via SMS [13]. In Austria's second mobile eID solution, the second factor of authentication was implemented via scanning a QR code with a mobile app [14, 15]. In Austria's third and newest mobile eID solution, authentication is done with a mobile app which uses a secure element (SE) on the mobile device for cryptographic operations [16]. Austria's example has been analysed extensively in the literature, both from architecture and legal perspectives [17, 18]. A novel approach to mobile authentication has been implemented in Baltics with the *Smart-ID* solution, where users authenticate with the *Smart-ID* app that is installed on their mobile devices [19, 20]. The *Smart-ID* solution architecture-wise is only partially server-based because the authentication scheme is based on a shared-RSA algorithm [21]. Hence server cannot generate a signature without the client and vice versa. The communication between the *Smart-ID* app and the server has been studied by intercepting network traffic via a Man-in-the-Middle (MitM) attack [22].

In SIM-based architecture, a security token is implemented via the subscriber identity module SIM (SIM), meaning that the user's private keys and cryptographic operations are handled by the SIM card (an enhanced SIM card is required) [23-25]. This architecture has been implemented in several countries, including Estonia, Lithuania, Iceland, Norway and

Switzerland [26]. However, Estonia's Mobile eID solution *Mobile-ID* has been more discussed in the literature compared to other solutions [6, 17, 27].

Finally, in device-based architecture, cryptographic operations are handled within mobile devices [28]. Today mobile devices are more powerful, and they can have a trusted execution environment (TEE), where cryptographic operations can be executed [29, 30]. This approach has been proposed in Brazilian and German mobile eID solutions [26, 29].

In the EU, there are several countries such as Bulgaria, Cyprus, Greece, Italy, Malta, Romania and Slovakia which does not have any kind of mobile eID solution in place [31]. In the literature, there can be found detailed information only about a few countries' mobile eID solutions, e.g., Germany [29], Austria [16] and Estonia [17, 19], and with such a fast-developing field as mobile technologies, previously made security assumptions have changed. For example, SMS based authentication is not considered secure enough anymore. There exists a research gap between what has previously been researched in the literature and the current mobile eID solutions.

# 3 *eParaksts mobile* from the user's perspective

*eParaksts mobile* is a Latvian mobile eID solution that has two core functionalities – authentication and electronic signing [32].

The authentication functionality allows *eParaksts mobile* users to prove their identity when receiving electronic services in Latvia [32].

The electronic signing functionality allows *eParaksts mobile* users to sign documents with qualified electronic signatures that are recognised under eIDAS regulation and have the same legal strength as handwritten signatures [3].

This chapter describes these core functionalities from the user's perspective.

## 3.1 Signing up for *eParaksts mobile*

Acquiring *eParaksts mobile* can be split into three parts – signing up for *eParaksts mobile* services, setting up the *eParaksts mobile* app on a mobile device to enable the authentication capability and finally, creating an electronic signature certificate by setting up a password for electronic signature creation [33].

### 3.1.1 Signing up for *eParaksts mobile* services

Signing up for the *eParaksts mobile* services requires that the user completes the following steps [33]:

1. Start the process by authenticating to the *mobile.eparaksts.lv* website using either the unified authentication system "*Latvija.lv*" (that supports bank authentication) or using an eID card or *eParaksts* card (see Figure 1). After successful authentication, the *mobile.eparaksts.lv* website obtains the user's legal name, surname and national identification number.

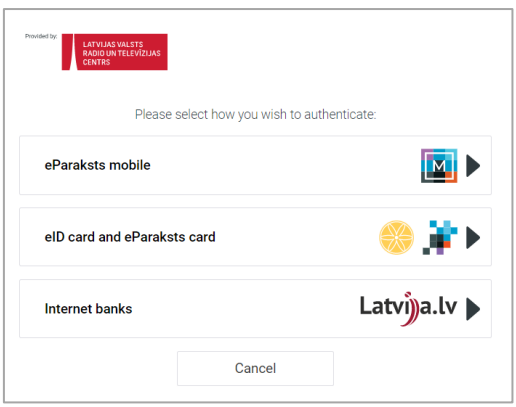


*Figure 1. Authentication options for authenticating to eparaksts.lv*

2. The user must provide their phone number to the *mobile.eparaksts.lv* website (see Figure 2). *eParaksts mobile* verifies the user's ownership of the phone number by sending an OTP code via SMS that the user must enter on the website within 60 seconds (see Figure 3). The phone number is used as an additional authentication factor.

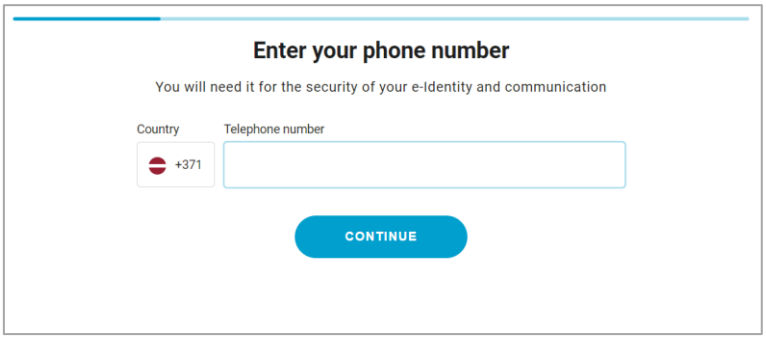


*Figure 2. The mobile.eparaksts.lv form for entering a phone number*

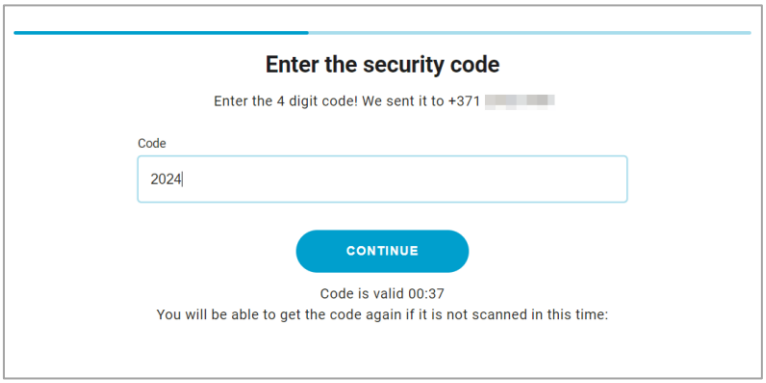


*Figure 3. The mobile.eparaksts.lv form for entering SMS OTP code*

3. The user must provide their email address to the *mobile.eparaksts.lv* website (see Figure 4). *eParaksts mobile* verifies the user's ownership of the email account by sending an OTP code via email that the user must enter on the website within 2 minutes (see Figure 5). Email address is used for sending informative emails from *eParaksts mobile*, such as creation and annulment of user's certificates.

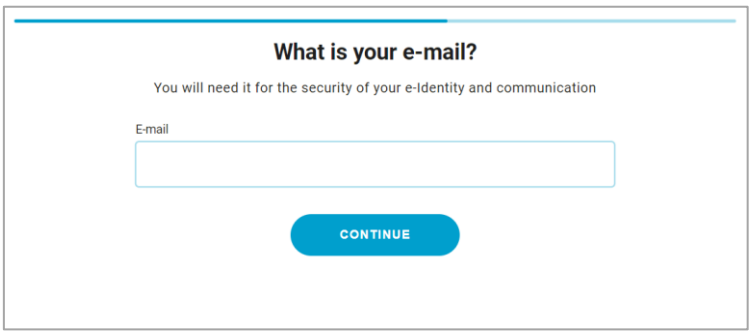


*Figure 4. The mobile.eparaksts.lv form for entering an email address*

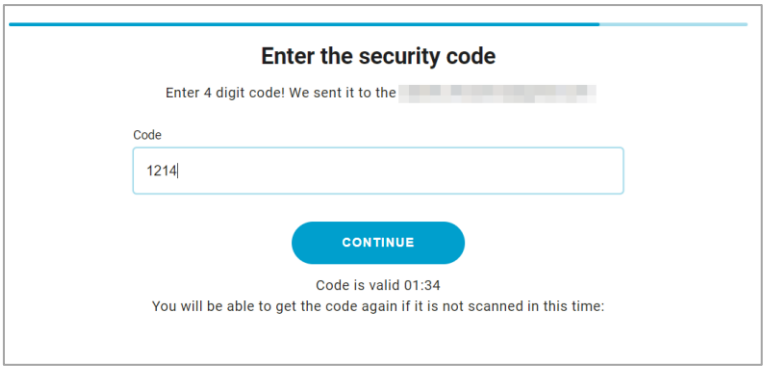


*Figure 5. The mobile.eparaksts.lv form for entering an email OTP code*

4. The user must provide data about their identification document (either passport or eID card), such as document number and expiration date, to the *mobile.eparaksts.lv* website (see Figure 6). This information is necessary for preparing the contract.

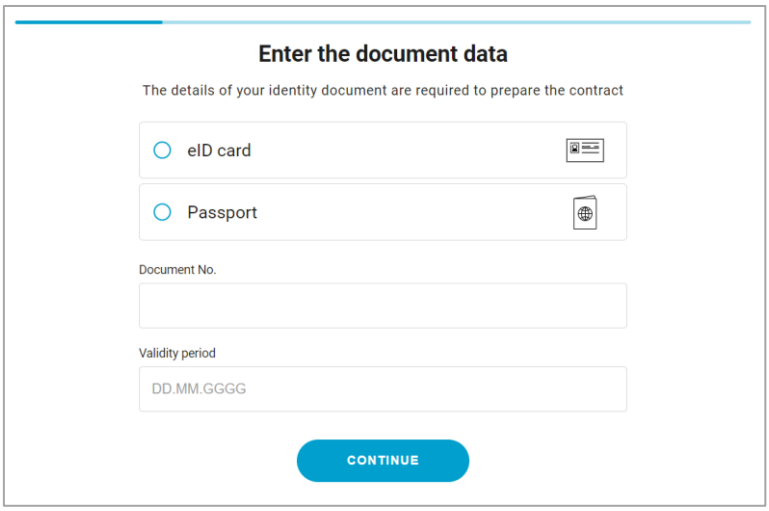


*Figure 6. The mobile.eparaksts.lv form for entering document data*

At this point, a contract is prepared between the user and the provider of *eParaksts mobile* – Latvia State Radio and Television Center (LVRTC). The contract does not have a termination date, but it can be cancelled by out-of-band means (e.g., by writing an email to *eParaksts* customer service). This contract contains a seven-digit User ID, e.g., "2428385", that is used for authentication with *eParaksts mobile*. It is possible to change the User ID to the user's national identification code (and back to the original User ID) on the *eParaksts* website (see Figure 12).

5. The user must sign the contract for receiving *eParaksts* services. The user can choose to sign the document electronically using their eID card or physically by giving a handwritten signature. In the latter case, the user can choose to have a courier deliver contract and validate their identity or visit a notary to sign the contract there.

### 3.1.2 Setting up the *eParaksts mobile* app

The following steps can be completed only after the contract has been signed and this fact has been processed by LVRTC. The user can have set up only one instance of the *eParaksts mobile* app at a time. To set up the *eParaksts mobile* app, the user needs to download and install the *eParaksts mobile* app from the Google Play Store[1] or Apple App Store[2] and complete the following steps [33]:

1. The user must create a 4-digit PIN code on their *eParaksts mobile* app and optionally add biometric credentials (fingerprint, *Touch ID* or *Face ID* for the latest *iPhone* models) authentication if their device supports it (see Figure 7). The option to authenticate with a PIN code remains active even if biometric authentication is enabled. There are no restrictions on what the 4-digit combination might be, yet the app displays a warning message for PIN codes that it considers insecure (such as "1234" or "0000") (see Figure 8). The PIN code (or biometric credentials, if enabled) is used for user authentication with *eParaksts mobile* (see Section 3.2.1 "Authentication").

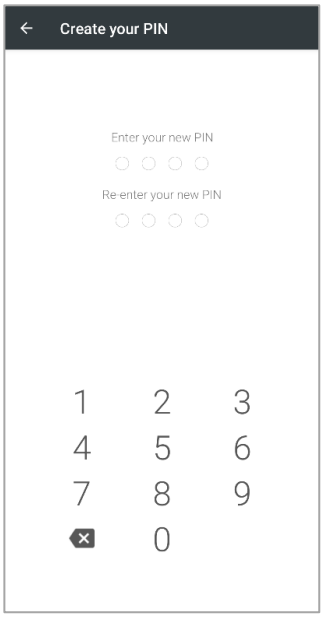


*Figure 7. The view for creating a PIN code in the eParaksts mobile app*

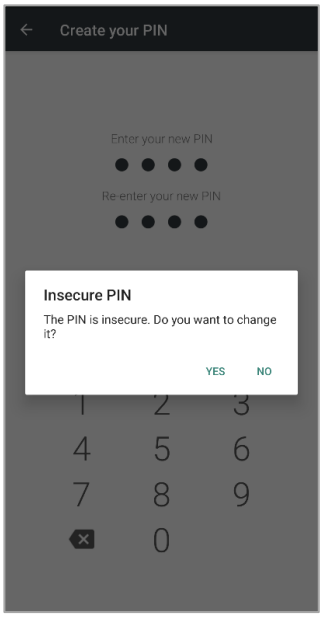


*Figure 8. The warning message in the eParaksts mobile app about insecure PIN*

[1] https://play.google.com/store/apps/details?id=lv.eparaksts.mobile

[2] https://apps.apple.com/lv/app/eparaksts-mobile/id1319846568

2. The user must enter in the app (see Figure 9) their *eParaksts mobile* User ID and authentication code that are displayed on the *eparaksts.lv* website as "User number" and "Application code", respectively (see Figure 10). The codes must be entered within 5 minutes. This can be entered either manually or by scanning a QR code on the website (see Figure 10). If the user does not have an active session with *eparaksts.lv*, step 1 must be completed again. This step allows linking the confirmed user identity on *eparaksts.lv* with the *eParaksts mobile* app.

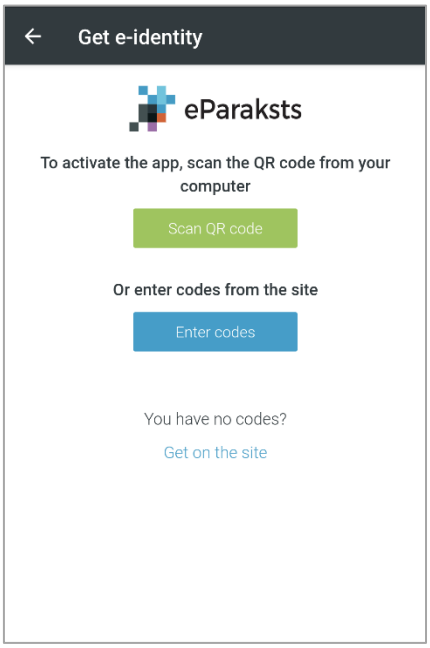


*Figure 9. The form for linking the user's identity to the mobile device in the eParaksts mobile app*

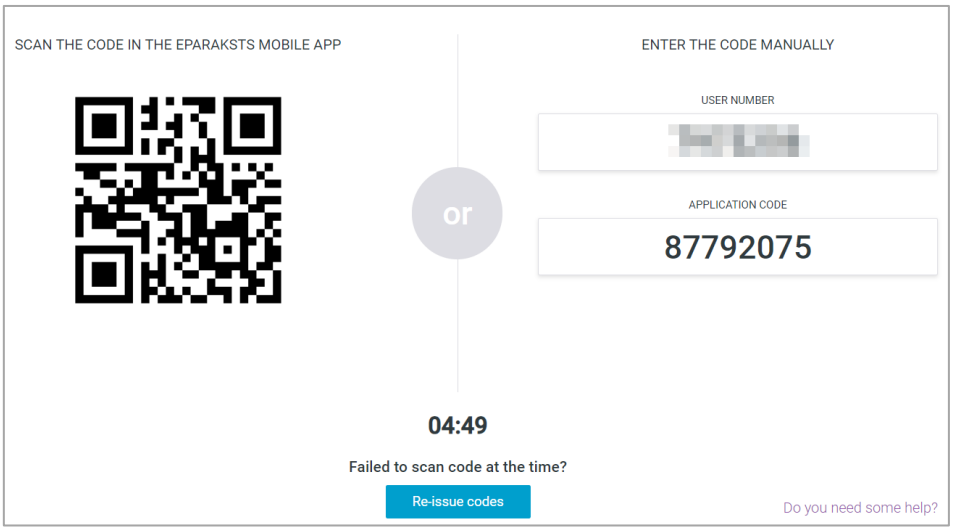


*Figure 10. The eparaksts.lv window displaying codes for linking the mobile app with the user's identity*

3. The user will receive an OTP code via SMS that should be entered into the *eParaksts mobile* app as a second-factor authentication (in this case, the first factor is the user's authenticated session on *eparaksts.lv*) (see Figure 11). The additional authentication factor protects against potential shoulder surfing attacks where the adversary might see the User ID and authentication code displayed on the screen and enter them before the legitimate user enters them on their mobile app.

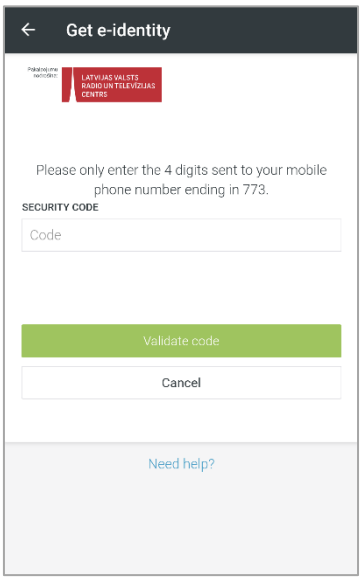


*Figure 11. The form for entering SMS OTP code in the eParaksts mobile app*

Now the user has created their authentication certificate and can use *eParaksts mobile* for authentication purposes. The issued certificate is valid for three years from the moment it has been issued. However, it is possible to cancel the certificate on the *eParaksts* website (see Figure 12) and repeat the steps described in this section to get a new one.

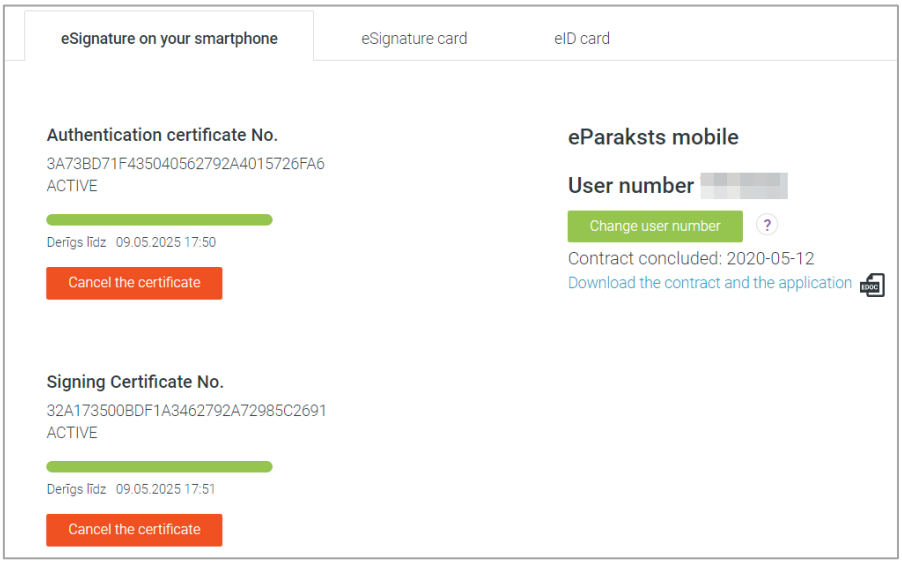


*Figure 12. Information about user's eParaksts mobile products on the eParaksts website*

After the *eParaksts* mobile app has been set up, it is possible to (a) change the PIN code, (b) enable or disable the authentication with biometric credentials and (c) delete the e-identity (authentication private key and certificate) from the device (see Figure 13):

(a) To change the existing PIN code, the user must enter their current PIN code or use the biometric credentials. After successfully providing the current credentials, the view for creating a PIN (see Figure 7) is displayed again.

(b) Enabling or disabling biometric credentials does not require any additional steps.

(c) When deleting the e-identity, a warning message "*Are you sure you want to delete your e-identity?*" is displayed. After confirming deletion, no additional authentication is required, and the authentication private key and certificate are deleted from the device.

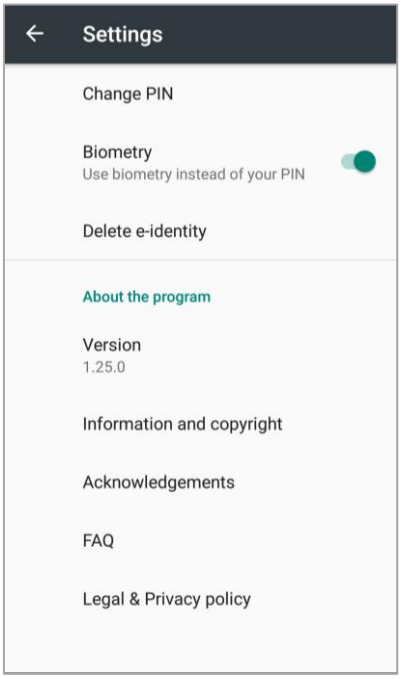


*Figure 13. Settings in the eParaksts mobile app*

### 3.1.3 Creating a certificate for electronic signature creation

Additionally, the user can set a password that is necessary for using the electronic signature functionality of *eParaksts mobile*. To set up a password for electronic signatures, the user needs to complete the following steps [33]:

1. To start the process, the user must authenticate to the *eparaksts.lv* website using either the unified authentication system "*Latvija.lv*" (that supports bank authentication) or using an eID card or *eParaksts* card, or *eParaksts mobile* solution (see Figure 1). After successful authentication, the *mobile.eparaksts.lv* website obtains the user's legal name, surname and national identification number.
2. The user must agree to the terms displayed on the *eparaksts.lv* website before continuing (see Figure 14), such as:
    (a) that the existing contract remains valid,
    (b) the General Terms and Conditions of the Trust Services [34],
    (c) receiving a qualified certificate that will be stored in a QSCD and used for qualified signature creation in accordance with the eIDAS regulation,
    (d) the service will be provided in accordance with the trust service "*eParaksts*" provision policy (OID: 1.3.6.1.4.1.32061.2.1.3.2),
    (e) a new certificate will be issued.

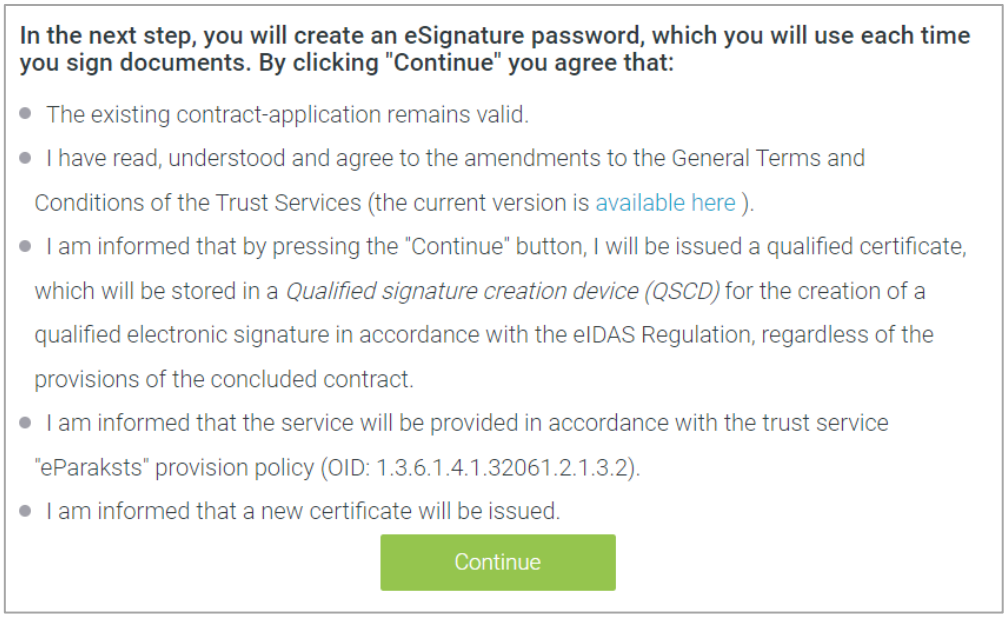


*Figure 14. The eparaksts.lv form for agreeing to displayed terms.*

3. An authentication request is sent to the user's *eParaksts mobile* app (see Figure 19). The user must confirm authentication to the *mobile.eparaksts.lv* website by entering the previously set PIN or biometric credentials (if enabled) (see Figure 22).
4. The user will receive an OTP code via SMS that should be entered into the *eparaksts.lv* website as a second-factor authentication (in this case, the first factor is the authentication with the *eParaksts mobile* app) (see Figure 15).

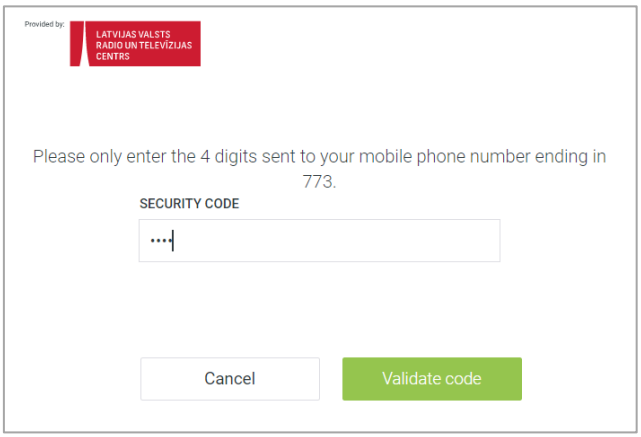


*Figure 15. The eparaksts.lv form for entering SMS OTP code*

5. The user must create a password on the *eparaksts.lv* website that must be at least six characters long (see Figure 16). This password will be used to authorise the electronic signature creation for the documents.

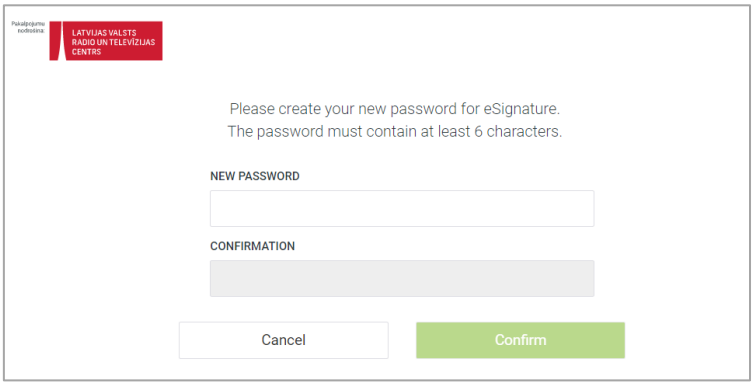


*Figure 16. The eparaksts.lv form for creating the signing password*

Now the user has created their electronic signature certificate and can use *eParaksts mobile* to sign documents. The user cannot change the password or suspend the certificate, but it is possible to cancel the certificate on the *eParaksts* website (see Figure 12) and do the steps described in this section to get a new one.

## 3.2 Using *eParaksts* mobile

### 3.2.1 Authentication

Using *eParaksts mobile* for authentication is possible on any website that has implemented either *eParaksts mobile* authentication directly or relies on the unified authentication system "*Latvija.lv*" (widespread identity federation solution in Latvia). When the user chooses to authenticate with *eParaksts mobile*, the authentication requests are forwarded to the *eParaksts mobile* website. This website, where the user wants to authenticate, can be accessed on any device, meaning that user can authenticate themselves regardless of whether they are using the same device on which the *eParaksts mobile app* is set up or a different device. However, if the user is accessing the website on the same device where the *eParaksts mobile* app is set up, then the first two steps from the process described below are skipped because the mobile app is launched without the user's intervention. The process goes as follows:

1. On the website, where the user wants to authenticate, the user selects authentication with *eParaksts mobile* as an authentication mechanism and enters their User ID when prompted (see Figure 17). Anyone who knows the user's User ID is able to initiate the authentication on the user's behalf. A waiting page is displayed on the website until the following steps are carried out (see Figure 18).

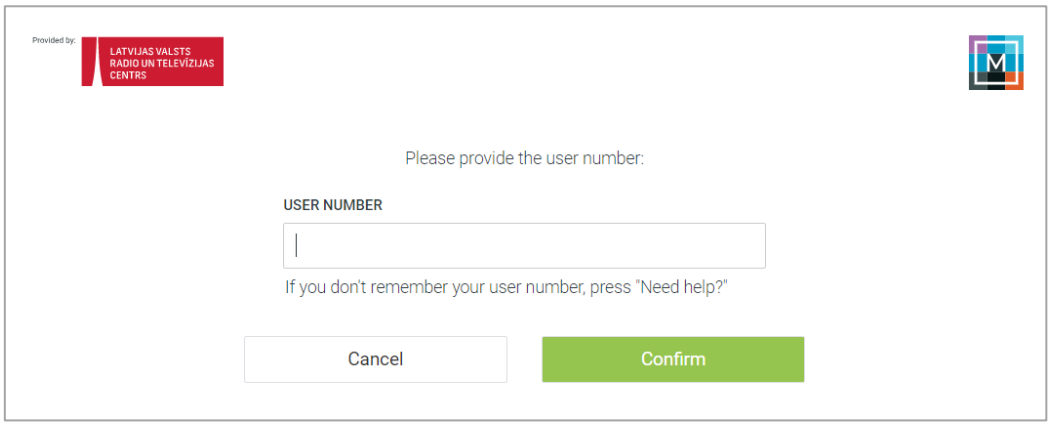


*Figure 17. The eidas.eparaksts.lv form for user authentication*

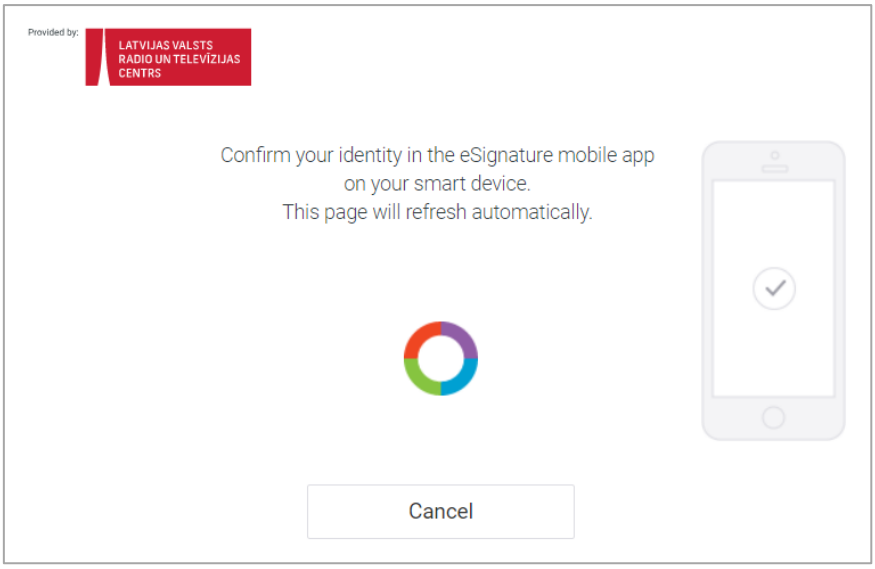


*Figure 18. A waiting page on the eidas.eparaksts.lv*

2. On their mobile device, the user receives a notification "Authentication. *eParaksts mobile* requests for identification" (see Figure 19). The user must click on the notification.

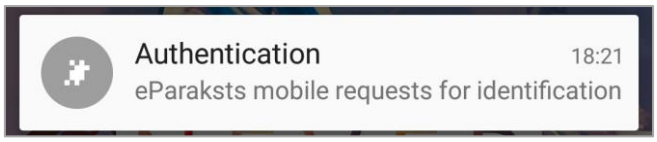


*Figure 19. The notification from the eParaksts mobile app for authentication*

3. The *eParaksts mobile* app is launched. There, the user must confirm or cancel the authentication request initiated by the website where the user is trying to authenticate (see Figure 20 and Figure 21). If the user cancels the request, the authentication flow gets terminated. Otherwise, if the user confirms the request, the following steps are carried out.

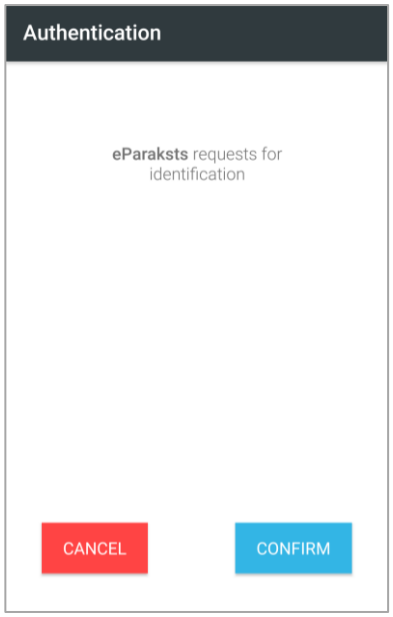


*Figure 20. The form for authentication request from eparaksts.lv in the eParaksts mobile app*


*Figure 21. The form for an authentication request from a different service (id2.rtu.lv) in the eParaksts mobile app*

4. The user must authenticate themselves to *eParaksts mobile* by entering the PIN code or biometric credentials (if enabled) (see Figure 22). The user has three attempts to enter the correct PIN code; otherwise, their certificate gets annulled and deleted from *eParaksts mobile*.

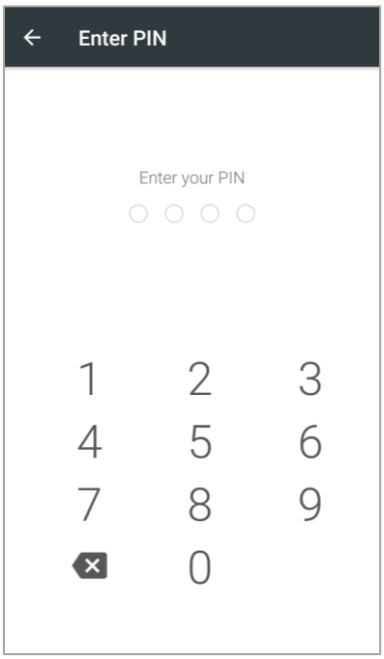


*Figure 22. the form for entering the PIN code in the eParaksts mobile app*

After successful authentication in the *eParaksts mobile* app, the user gets authenticated to the website they wish to use. If the user wants to authenticate to an e-service provider other than *eParaksts*, then prior to authenticating to the target website, the user must give explicit consent to share their personal data with that e-service provider. For this reason, a consent form in the browser is displayed to the user (see Figure 23).

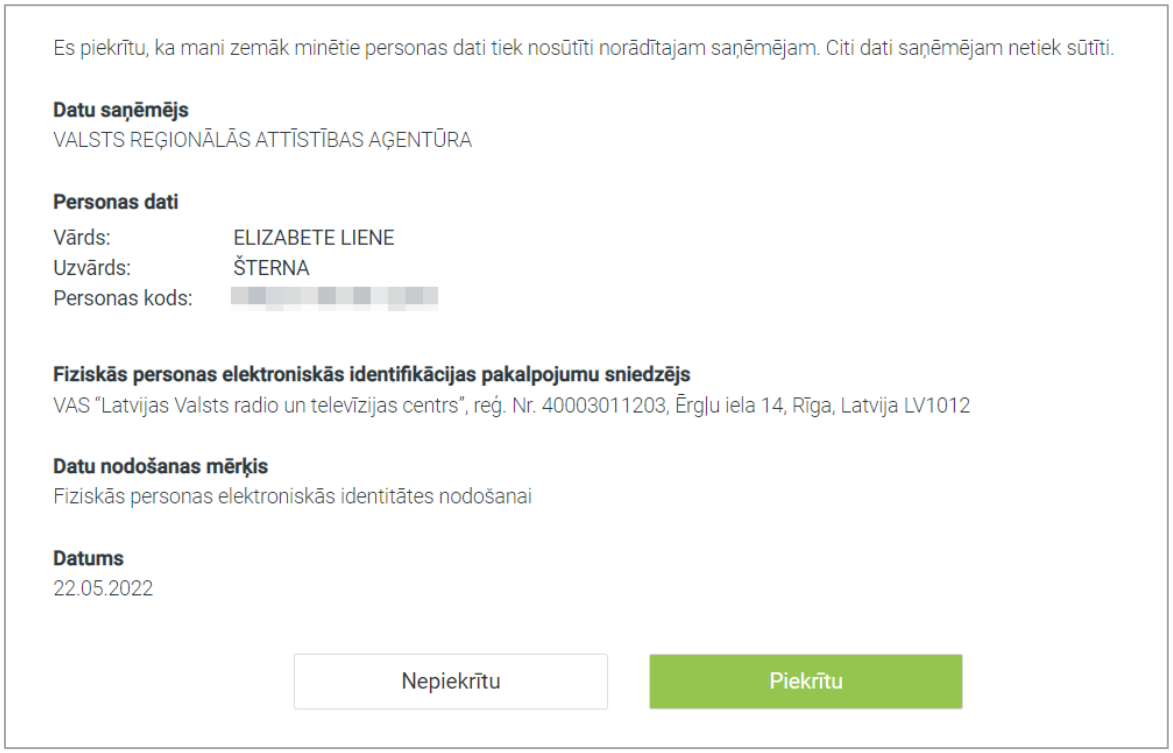


*Figure 23. A consent form for sharing the user's personal data with another e-service provider (in Latvian)*

### 3.2.2 Electronic Signing

Electronic signing can be done either on the *eparaksts.lv* website or on the *eParakstsLV* [3] mobile app (note – this is a separate app just for electronically signing documents and is out of the scope of this thesis). Either way, the workflow is as follows:

1. The user uploads the document they wish to sign to *eparaksts.lv* website (or *eParakstsLV* app) and starts the signing process (see Figure 24).

[3] https://play.google.com/store/apps/details?id=lv.eparaksts.signer

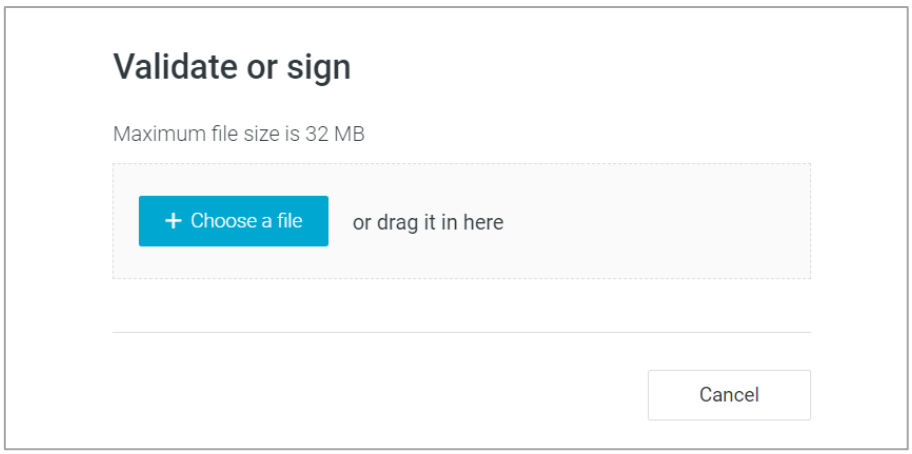


*Figure 24. eparaksts.lv form for uploading a file to be signed*

2. On the website (or *eParakstsLV* app), the user selects the option to sign with the *eParaksts mobile* app (see Figure 25) and is prompted to provide their User ID on the website (see Figure 17).

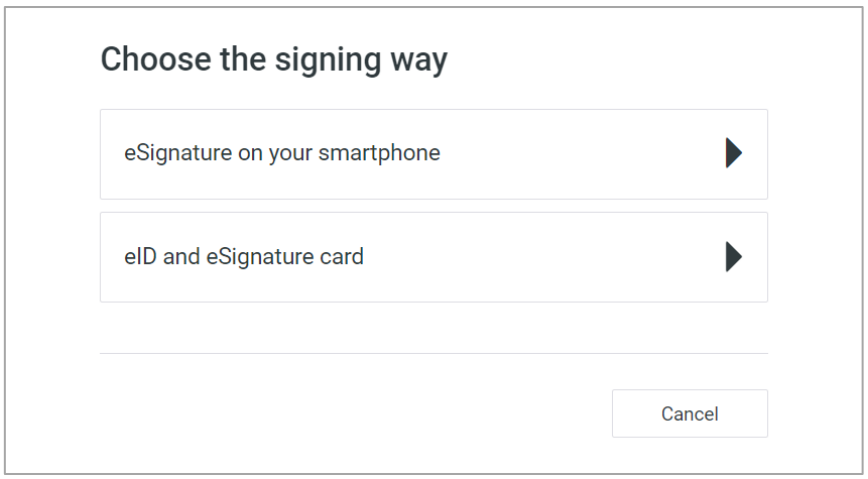


*Figure 25. eparaksts.lv form for choosing to sign with the eParaksts mobile app*

3. The user receives a notification "Authentication. *eParaksts mobile* requests for identification" on their mobile device (see Figure 19), and the user must click on the notification.
4. The *eParaksts mobile* app is launched, and there the user must confirm or cancel the authentication request (see Figure 20). If the user cancels the request, the authentication flow gets terminated. Otherwise, if the user confirms the request, the following steps are carried out.

5. The user must authenticate themselves to *eParaksts mobile* by entering the PIN code or biometric credentials (if enabled) (Figure 22). The user has three attempts to enter the correct PIN code; otherwise, their certificate gets annulled and deleted from *eParaksts mobile*.
6. After successful authentication in step 5, on the *eparaksts.lv* website (or *eParakstsLV* app) where the user previously uploaded the document, the user is prompted to provide their signature password (see Figure 26). The user has 15 attempts to enter the correct password. After that, the access to the signature certificate is denied, but the certificate itself remains active. In order to restore the electronic signing capability, the user must manually annul the electronic signature certificate via the *eParaksts* website (Figure 12) and create a new certificate following the steps described in Section 3.1.3 “Creating a certificate for electronic signature ”.

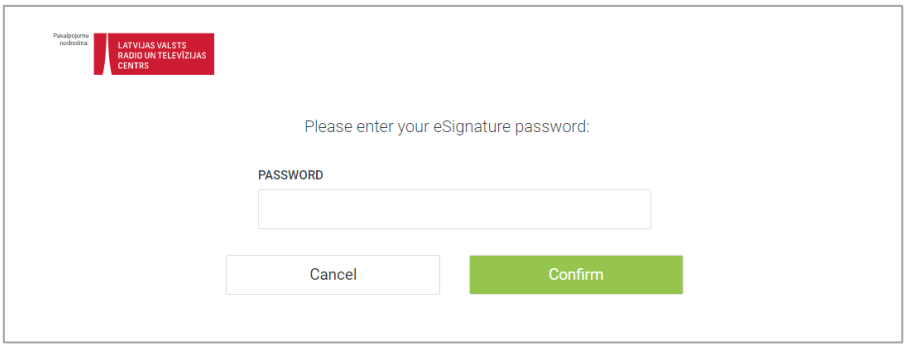


*Figure 26. eidas.eparaksts.lv form for entering signature password*

After successful password validation, the document is signed and available to the user for downloading (see Figure 27).

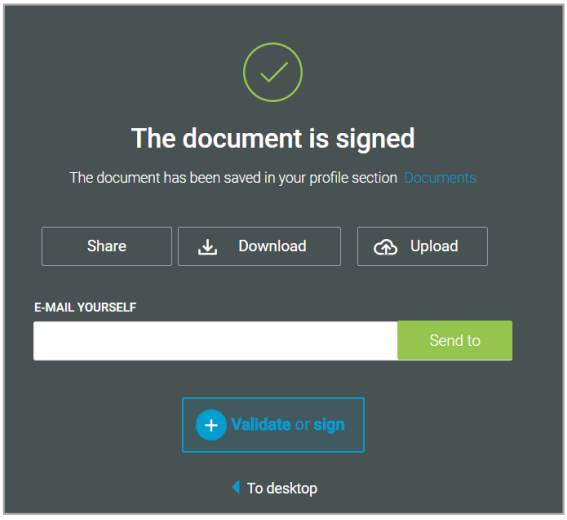


*Figure 27. eparaksts.lv form after the document has been signed*

# 4 Intercepting network traffic of *eParaksts* mobile

To understand how the *eParaksts mobile* authentication and electronic signing schemes have been implemented, inspecting the network traffic between the *eParaksts mobile* app and server is necessary. The network traffic is protected by Transport Layer Security (TLS) protocol to mitigate risks associated with Man-in-the-Middle (MitM) attacks [35]. TLS is a security protocol that ensures authentication of parties involved in the communication, encryption and integrity of the transmitted data [36]. The first step of establishing a TLS session is performing a TLS handshake, during which the server proves its identity to the client using the server's TLS certificate, both parties agree on cypher suites and establish session keys for message encryption [36].

Furthermore, we discovered that certificate pinning is used in the *eParaksts mobile* app as protection against MitM attacks [37]. The *eParaksts mobile* app has hardcoded information about the *eParaksts mobile* server's TLS certificate, and a successful TLS session will be established only if the server's certificate that is received during the TLS handshake phase matches the hardcoded information in the mobile app [38].

To obtain the plaintext communication between the *eParaksts mobil*e client (mobile app and browser) and the *eParaksts* server, we implemented and performed a successful MitM attack. In a TLS MitM attack, a proxy is used to intercept and forward the network traffic between communicating parties. The proxy establishes two new connections (one with the client and one with the server), allowing the proxy to decrypt and forward the messages sent between the client and the server [39]. During the TLS handshake phase, the proxy offers its TLS certificate to the client to establish a TLS session. Since the *eParaksts mobile* app uses certificate pinning, it normally would reject such a session with the proxy. Therefore it is necessary to modify the app such that it would trust the fraudulent certificate provided by the proxy, and a TLS session could be established.

To sum up, a successful execution of the MitM attack requires two things:

1. Setting up an environment for traffic interception. For this, two testbeds will be configured.
2. Modification of the *eParaksts mobile* app to disable certificate pinning.

## 4.1 Setting up an environment for traffic interception

The first step in the environment setup for the MitM attack is setting up a proxy that will intercept the traffic. The chosen tool for traffic interception is *Mitmproxy*[4]. We chose the following modes of operation of the *Mitmproxy* proxy [40]:

1. Regular – the client must be configured to use the *Mitmproxy* proxy.
2. Transparent – there is no need to configure the client because the network traffic will be redirected to the *Mitmproxy* proxy at the network layer. The proxy is invisible to the client.

The transparent mode must be used to capture the traffic between the *eParaksts mobile* app and the *eParaksts* server. At the same time, the regular mode must be used to capture the traffic between the *eParaksts* website and the *eParaksts* server. Intercepting the network traffic of the mobile app and the website is necessary because both clients are used in authentication and electronic signing processes. Therefore, two separate testbeds must be set up. The setup of the testbeds is described in the following sections.

[4] https://mitmproxy.org/

### 4.1.1 Testbed for intercepting network traffic between the *eParaksts mobile* app and the *eParaksts* server

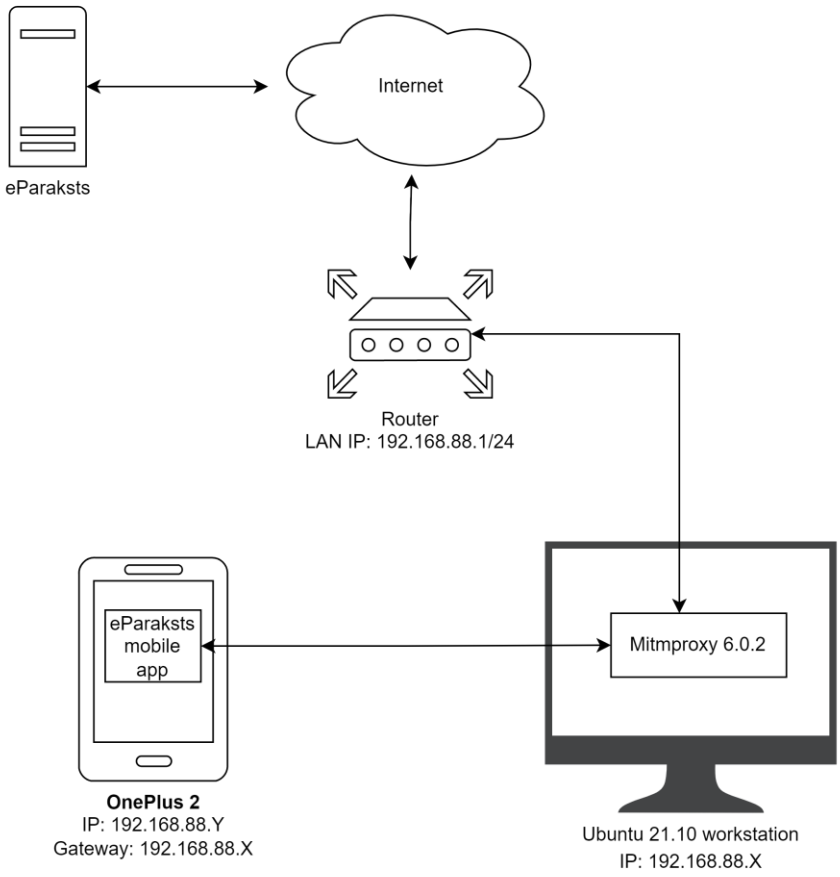


*Figure 28. Testbed for intercepting network traffic between the eParaksts mobile app and the server*

The first testbed (see Figure 28) is necessary for intercepting network traffic between the *eParaksts mobile* app and the *eParaksts* server. Mitmproxy's transparent mode must be used for intercepting network traffic on Android apps because many Android apps (including the *eParaksts mobile* app), by default, ignore the network proxies configured in the Android system settings [41].

According to *Mitmproxy* documentation [42], the transparent mode is only available on *Linux* or *macOS* operating systems. Therefore, a workstation with *Ubuntu 21.10* OS was used. *Mitmproxy* v6.0.2 was installed on this machine. The network on this workstation, as per *Mitmproxy* documentation [42], was configured as follows:

1. IP forwarding was enabled to ensure that the machine forwards network packets,

```
sysctl -w net.ipv4.ip_forward=1
sysctl -w net.ipv6.conf.all.forwarding=1
```

2. ICMP redirects were disabled. This is necessary because otherwise, the machine would inform the test device (a mobile device with the *eParaksts mobile* app) that a shorter route is available by skipping the proxy,

```
sysctl -w net.ipv4.conf.all.send_redirects=0
```

3. Following iptables rules were created to redirect the necessary network traffic to the *Mitmproxy* proxy.

```
iptables -t nat -A PREROUTING -i ens33 -p tcp --dport 80 -j REDIRECT --to-
port 8080
iptables -t nat -A PREROUTING -i ens33 -p tcp --dport 443 -j REDIRECT --to-
port 8080
ip6tables -t nat -A PREROUTING -i ens33 -p tcp --dport 80 -j REDIRECT --to-
port 8080
ip6tables -t nat -A PREROUTING -i ens33 -p tcp --dport 443 -j REDIRECT --to-
port 8080
```

The test device was a *OnePlus 2* smartphone with *Android 6.0.1* OS (*OxygenOS 3.6.1*), on which the modified *eParaksts mobile* app v1.25.0[5] was installed (see Section 4.2 "Modification of the *eParaksts mobile* app"). To install mobile apps that do not come directly from *Google Play Store*, a toggle "Unknown apps" that allows installing apps from unknown sources must be enabled in *Settings → Security & Fingerprint*. To ensure that network traffic was routed through the *Mitmproxy* proxy, the mobile test device (*OnePlus 2*) was connected to the same LAN to which the machine running *Mitmproxy* was connected (*Ubuntu 21.10*). In order to force all network traffic to be routed through the proxy, the network settings for the mobile device were configured to set the machine running *Mitmproxy* as a gateway (see Figure 29).

[5] When the network traffic analysis was performed, v1.25.0 was the newest version of the *eParaksts mobile* app. Since then, newer *Android* mobile app versions have been released that do not support *Android 6* anymore. The latest *eParaksts mobile* app requires at least *Android 8.0*.

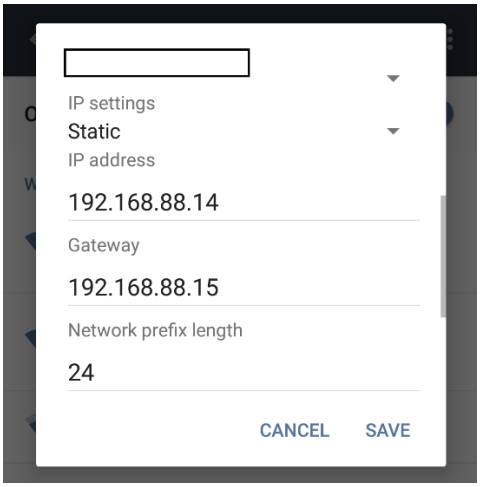


*Figure 29. WiFi network settings on a mobile device*

Additionally, it was necessary to install the *Mitmproxy* CA certificate on the mobile test device. This forces the device to trust *Mitmproxy's* built-in certificate authority. This makes the *eParaksts mobile* app accept *Mitmproxy's* certificate. *Mitmproxy* generates the CA certificate whenever the *Mitmproxy* proxy is run and can be downloaded from the locally served *mitm.it* website[6] [43]. The CA certificate can be installed on the test device by navigating to *Settings* → *Security & Fingerprint* → *Advanced* → *Credential Storage* → *Install from storage* and selecting the downloaded certificate.

The *Mitmproxy* was started with the following command:

```
mitmweb --mode transparent –showhost
```

[6] http://mitm.it/

### 4.1.2 Testbed for intercepting network traffic between the *eParaksts website* and the *eParaksts* server

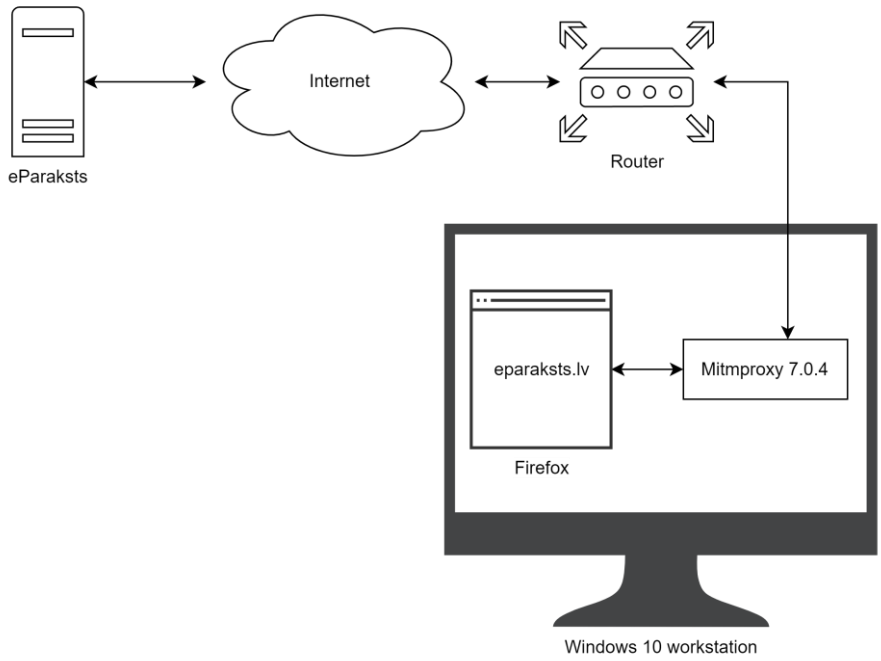


*Figure 30. Testbed for intercepting network traffic between the eParaksts mobile website and the server*

The second (see Figure 30) testbed is necessary for intercepting the network traffic between *the eParaksts* website and the server. Many tools, for example, a browser's built-in developer tools, can be used to intercept browser network traffic because requests are not yet encrypted, but the responses are already decrypted when they are processed by the browser.

Nevertheless, a proxy can also be used to intercept this type of network traffic. *Mitmproxy* in regular mode was also used for traffic interception in this use case, since it offers a rich interface and was used in the first testbed. On *Windows 10* workstation, *Mitmproxy v.7.0.4* was installed. On the same *Windows 10* machine, the *Firefox* browser was used to access the *eParaksts* website. A manual proxy configuration was used in the *Firefox* network settings: HTTP/S Proxy was set to *127.0.0.1:8080*. The IP address *127.0.0.1* is used to establish a network connection to the same machine on which Firefox is used, and port *8080* is the default port on which *Mitmproxy* proxy is listening. This configuration ensures that all network traffic related to *Firefox* is redirected through the *Mitmproxy* proxy.

The *Mitmproxy* was started with the following command:

```
mitmweb
```

## 4.2 Modification of the *eParaksts mobile* app

As mentioned above, the *eParaksts mobile* app uses certificate pinning to mitigate risks associated with MitM attacks. Therefore, the mobile app must be modified to trick the app into accepting *Mimtproxy's* certificate and establishing a successful TLS session with the *Mitmproxy* proxy.

For this reason, an APK file (Android Package file with file extension *apk* is the file format used by *Android* OS for *Android* apps) of the *eParaksts mobile* app v1.25.0 was downloaded from *apkpure.com*[7]. *VisualStudio Code Extension APKLab*[8] was used to aid in making necessary modifications to the downloaded APK file. This tool offers APK decompilation, preparing for HTTPS inspection (disables certificate pinning), and rebuilding and signing the APK file, which is needed for this use case [44].

After decompiling the APK file with *APKLab,* it is possible to see the app's source code in *Dalvik* bytecode displayed in *Smali*. The source code is not obfuscated; therefore can be inspected without additional efforts to de-obfuscate. *APKLab* function "Prepare for HTTPS inspection" [44] automatically finds code that enables certificate pinning and modifies it. Following changes were made in the app's source code by *APKLab* (see Appendix I "Modifications of the *eParaksts mobile* app by *APKLab*" for code snippets):

1. Contents of *~/smali_classes/okhttp3/CertificatePinner.smali* method *check(String, java.util.list)* were commented out.
2. Contents of *~/smali_classes/okhttp3/CertificatePinner.smali* method *check$okhttp(String; Function)* were commented out.
3. Contents of *~/ smali_classes/okhttp3/internal/tls/OkHostnameVerifier.smali* method *verify(String, javax.net.ssl.SSLSession)* were commented out.
4. Network security configuration file generated with lax network configuration (this modification was unnecessary since the original configuration file was lax enough).

[7] https://apkpure.com/eparaksts-mobile/lv.eparaksts.mobile/versions

[8] https://marketplace.visualstudio.com/items?itemName=Surendrajat.apklab

Additionally, the *eParaksts mobile* app has a method that checks whether the source code has not been tampered with. We disabled this check by modifying a method *isTamperDetected()* (see Figure 31).

```
299  .method private isTamperDetected()Z
300      .locals 1
301      const/4 v0, 0x0
302      return v0
303      #Integrity check disabled
304      #.locals 1
305      #.annotation system Ldalvik/annotation/Throws;
306      #    value = {
307      #        Lcom/safelayer/mobileidlib/splash/SecurityHelper$TamperDetectionException;
308      #    }
309      #.end annotation
310      #
311      #.line 55
312      #invoke-direct {p0}, Lcom/safelayer/mobileidlib/splash/SecurityHelper;->isSignatureError()Z
313      #
314      #move-result v0
315      #
316      #return v0
317  .end method
```

*Figure 31. Modifications in method isTamperDetected() located in ~/smali_classes2/com/safelayer/mobileidlib/splash/SecurityHelper.smali*

Lastly, the APK file contained a root certificate of *eParaksts* "*DigiCertGlobalRootCA*". We discovered that this certificate needed to be replaced with a CA certificate generated by *Mitmproxy*. Otherwise, a successful TLS session could not be established.

After all modifications, the *APKLab* function "Rebuild the APK" is used to rebuild the APK file and sign it [44].

# 5 Analysis of the intercepted *eParaksts mobile* network traffic

To get a complete picture of how authentication and electronic signing processes work in *eParaksts mobile*, the network traffic was intercepted and analysed for 4 use case scenarios:

1. Setting up the *eParaksts mobile* app as described in Section 3.1.2 "Setting up the *eParaksts mobile* app",
2. Authentication with *eParaksts mobile* as described in Section 3.2.1 "Authentication",
3. Electronic signature creation as described in Section 3.1.3 "Creating a certificate for electronic signature ",
4. Electronic signing with *eParaksts mobile* as described in Section 3.2.2 "Electronic Signing".

For the network traffic interception, *Mitmproxy* was used as described in Section 4.1 "Setting up an environment for traffic interception". Both testbeds were used in parallel to get a full overview of the network traffic in the aforementioned use cases.

The captured traffic[9] is analysed in subsequent sections. All captured traffic takes place over HTTPS. Throughout requests and responses, authentication cookies are used. Request and response pairs such as repeated requests and requests related to visual information have been left out of the diagrams to maintain clear overview of the process.

[9] *Mitmproxy* files containing the network traffic captured in the course of this research are available at: https://github.com/elisterna/Mitmproxy_captures. Some personal information has been modified via Mitmweb UI.

## 5.1 Setting up the *eParaksts mobile* app

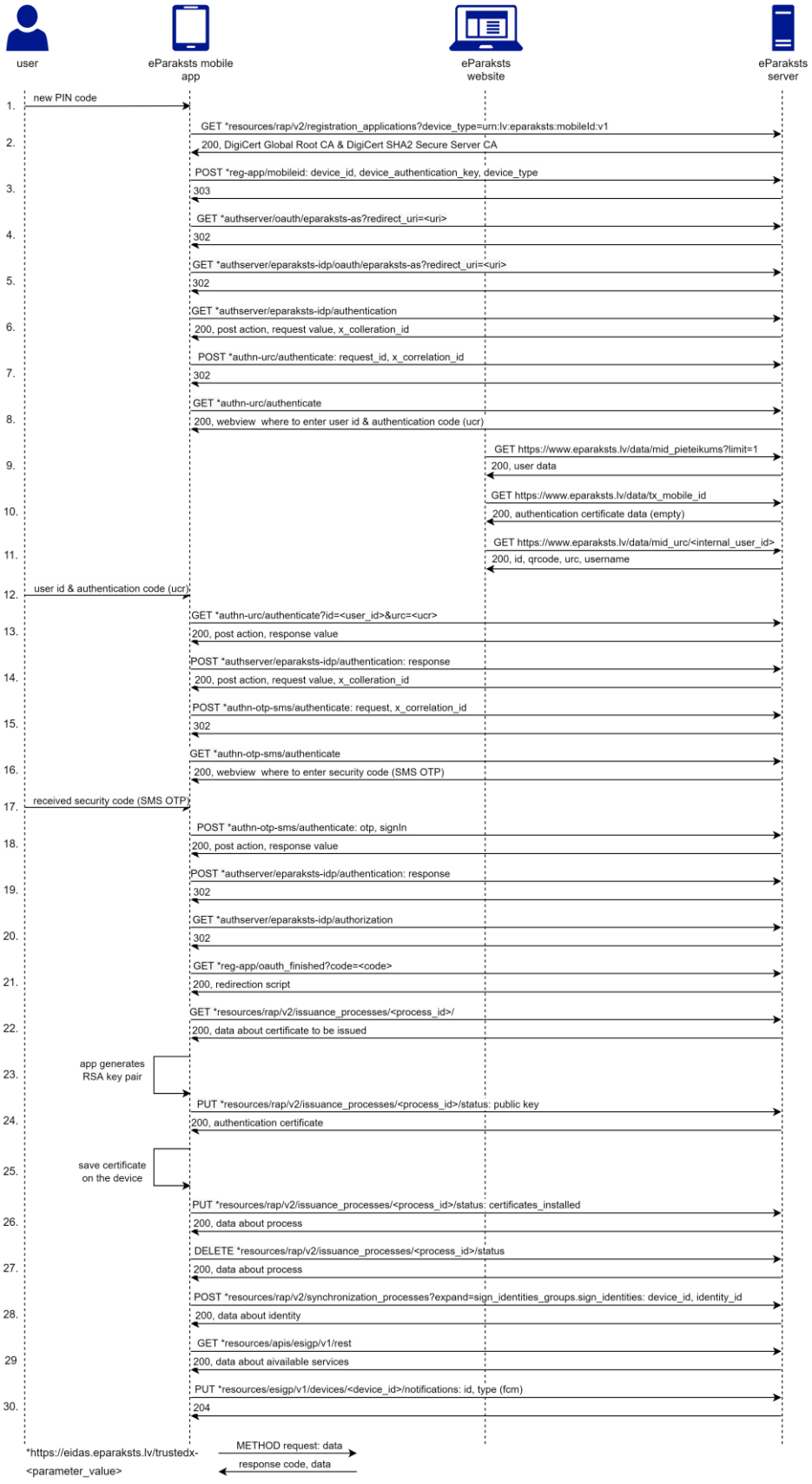


*Figure 32. A sequence diagram depicting the process of setting up the eParaksts mobile app*

Figure 32 depicts the process of setting up the *eParaksts mobile* app. The involved parties in this process are (1) the user who has registered for *eParaksts mobile* services, (2) the *eParaksts mobile* app that the user wants to set up, and (3) the *eParaksts* website (*www.eparaksts.lv*), where the user must be authenticated via provided authentication mechanisms, and (4) the *eParaksts* server (*eidas.eparaksts.lv*). In this scenario, the user has an active session with the *eParaksts* website.

This process can be split down into the following parts:

1. **Device registration.**

    1.1. The process begins with the user creating a PIN code for their e-identity on the app (step 1) that the user will use to confirm authentication with *eParaksts mobile* in the future. The PIN code is stored locally on the device. The app initiates the registration process using a JWS (JSON web token with a signature) token as the authorisation method (step 2), and the server responds with a URL for device registration. JWS signature is generated with an HS256 algorithm which means that the integrity of the signature is protected using an HMAC with a 256-bit symmetric key (the key is sent to the server in step 3). Decoded JWS header and payload:

```
Header:
{
  "typ": "sfly-device-auth-token",
  "alg": "HS256"
}
Payload:
{
  "nonce": "VgHu6Rugjf/dkbPPah9w3yyRikM=",
  "iat": 1647264721,
  "iss": "be4a620d46cfe6bab839b84ad147a991f8f83523622fbe4b92b9b488380db0a7"
}
```

    1.2. Then the app opens a *WebView* and sends the device ID, device authentication key and device type to the URL provided in the step 2 response (step 3). The device ID always stays the same and matches the issuer (*iss* field) of the JWS token in step 2. The app generates the device authentication key, which is used to sign the JWS token in step 2. For every registration attempt, a new key is generated. An example of the device ID, device authentication key and the device type:

```
device_id: be4a620d46cfe6bab839b84ad147a991f8f83523622fbe4b92b9b488380db0a7
device_authentication_key: cP1QYOo4RwCNjgewezJt0iEVVb/bSyksl80AZWt46ss=
device_type: urn:lv:eparaksts:mobileId:v1
```

**2. Linking mobile device to user's *eparaksts.lv* profile.**

For user authentication, *OAuth 2.0* authorisation code grant is used, and the user is authenticated with two-factor authentication (2FA).

2.1. After device registration, the app is redirected to the authentication page (steps 4 – 6), where the user will link the device with their *eparaksts.lv* profile (see Figure 9). To ensure that the user is authenticated on the same device that was registered in part 1, in step 6 server's response contains the subsequent request (method and URL), unique request and correlation identifiers that must be included in the next request. The mobile app can access the authentication page only after the server validates these values (step 8).

2.2. On a web browser where the user has already been authenticated to the *eparaksts.lv* website, the user is provided with their User ID and authentication code on the website (see Figure 10). Before displaying these codes (step 11), the browser requests to the server the user's authentication certificate (this certificate is created when the *eParaksts mobile* app is set up) to ensure that the user already does not have one (step 10). If a user had the authentication certificate, the website would display that this step has already been completed, and the process flow would be terminated.

2.3. The user submits the User ID and authentication code on the mobile app's authentication page (step 12), which are sent over to the server and validated to ensure that the values submitted by the user match the ones displayed in step 11 (step 13). This concludes user authentication with the first factor.

2.4. To ensure that second-factor authentication (SMS OTP) happens on the same registered device, i.e., the user does not try to continue this process from a different device, the following server's responses contain unique values that must be included in subsequent requests (steps 13 – 15).

2.5. The user is presented with a *WebView* where to submit an SMS OTP code (see Figure 11) on the mobile app (step 16). An SMS with the OTP code is sent to the phone number that was provided during signing up for *eParaksts mobile*. The device with the *eParaksts mobile* app is not physically linked to the SIM card, which indicates that the SIM card is not involved in authentication certificate issuance. The user submits the SMS OTP code in the app (step 17), which is sent over to the server and validated to ensure that this value matches the one in the SMS (step 18).

**3. Certificate issuance process.**

3.1. After the user has been authenticated with 2FA, the issuance process for the authentication certificate can be initiated. Prior to that, it must be ensured that the certificate is issued on the same device where the user was authenticated. Thus, the server's response (step 18) contains a unique value that must be included in the subsequent request (step19) that is followed by multiple redirects (steps 19 – 21) until the endpoint for the issuance process is reached (step 22). From this point on, the WebView is closed, and the mobile app includes JWS tokens in the authorisation header of the requests.

3.2. The *eParaksts mobile* app receives information about the issuance process, including configuration information for the authentication certificate's keys (key size and algorithm) (step 22), and generates a 2048-bit RSA key pair on the device (step 23). The *eParaksts mobile* app sends the public key to the server, and the server issues the authentication certificate to the user and sends the certificate to the mobile app (step 24). The certificate is then installed on the *eParaksts mobile* app (step 25), and the process is finished (step 27).

3.3. In order to synchronise data with the server, the *eParaksts mobile* app sends the device ID and "*sign_identities_groups.id*" that uniquely identifies the user's authentication certificate (step 28). In return, the server responds with a JSON structure of the "sign identity" (see Figure 42). The payload of the request made by the mobile app:


```
{
"device_id": "be4a620d46cfe6bab839b84ad147a991f8f83523622fbe4b92b9b488380db0a7",
"sign_identities_groups": [
      {
         "id": "vr73rpe5j7aeut7u0d783v5b9e412acr"
      }
   ]
}
```

**4. Device configuration.**

4.1. After the authentication certificate has been issued, the *eParaksts mobile* app can discover relevant API services (step 29) and set up a notification service for the authentication notifications in the future (step 30).

## 5.2 Authentication with *eParaksts mobile*

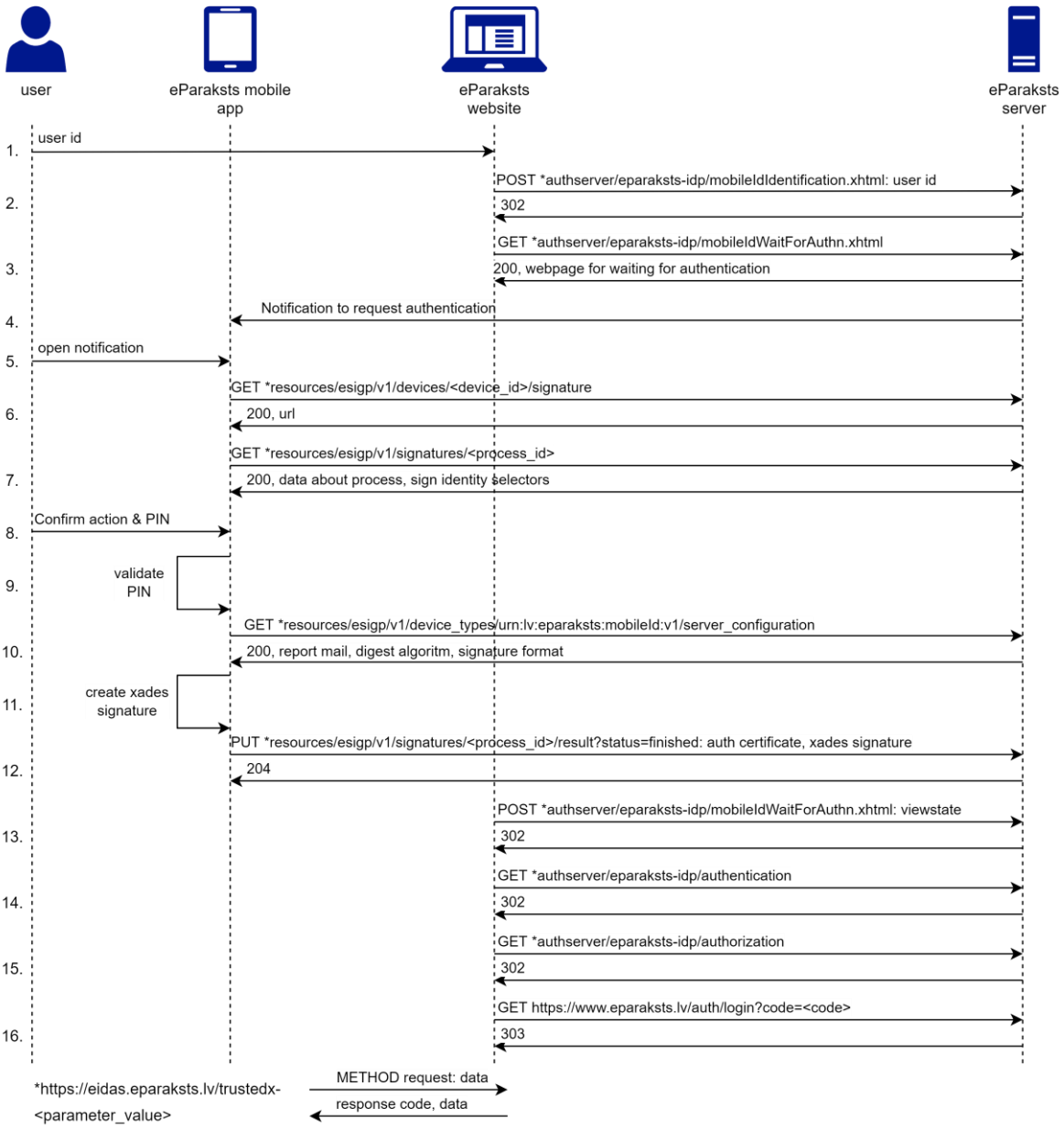


*Figure 33. A sequence diagram depicting the process of authentication with eParaksts mobile*

Figure 33 depicts the process of authentication with *eParaksts mobile* on the *eParaksts* website from the point where the user has already chosen *eParaksts mobile* as the authentication method. The involved parties in this process are (1) the user with the *eParaksts mobile* app set up on their device, (2) the *eParaksts mobile* app, (3) the *eParaksts*

website (*www.eparaksts.lv*), where the user wants to be authenticated, and (4) the *eParaksts* server (*eidas.eparaksts.lv*). The *eParaksts mobile* app includes in its requests sent to the server a JWS token signed with the device authentication key in the authorisation header. This process can be split down into the following parts:

1. **Initialisation.**
   1.1. The process begins with the user providing their User ID on the authentication page of the *eParaksts* website *eidas.eparaksts.lv* (see Figure 17) (step 1), which is sent to the server (step 2). The browser is redirected to a waiting page (see Figure 18) until the authentication described in part 2 is completed (step 3).
2. **Authentication.**
   2.1. The server sends a notification (see Figure 19) to the user's *eParaksts mobile* app (step 4). When the user opens the notification (step 5) mobile app tries to initiate the authentication process, and if there is an authentication request, the server responds with a URL for that request (step 6). The mobile app creates a request to the provided URL and receives information about the authentication request, such as information about the requesting service (the website where the user is trying to authenticate), process ID, domain (in this case, the value is "citizen"), user's authentication certificate, operation type (in this case it is authentication) and the endpoints for completing authentication request (step 7). The domain is presumably used to differentiate between natural and legal entities. There are three endpoints for completing the authentication request. One is for successful authentication, one is for cancelled authentication (either due to cancelling the request or failing to enter the correct PIN 3 times in a row), and one is for failed authentication attempts due to technical reasons.
   2.2. Information about the requesting service is displayed to the user with a prompt to confirm or cancel authentication (see Figure 20). When the user confirms the authentication request, the user must provide the PIN code (step 8). The *eParaksts mobile* app locally validates the PIN code (step 9) and, upon successful validation, proceeds to the next steps.

2.3. The *eParaksts mobile* app signs the challenge (consists of process ID, operation type, domain and requesting service) with the user's private key to create an XAdES signature (see Figure 34). Before a signature can be generated, the *eParaksts mobile* app requests configuration information such as signature format and digest algorithm (step 10). Then XAdES signature is created on the *eParaksts mobile* device (step 11) and sent over to the server (step 12).

```
<Content Id="Service" MimeType="text/xml">
    <Service Domain="urn:safelayer:eidas:domain:oauth:client" Id="trustedx-reg-app" Name="eParaksts"/>
</Content>
<Content Id="SignatureProcess" MimeType="text/xml">
    <SignatureProcess Domain="citizen" Id="6f690coav9b6rs5jn31ognr4vg" OperationType="authentication"/>
</Content>
```

*Figure 34. A snippet of a portion of the XAdES signature that contains the main part that is signed in the authentication process*

**3. Processing authentication.**

3.1. The server validates the XAdES signature and authenticates the user by responding to the web browser's repeated requests with a redirect (step 13). In those requests, the web browser repeatedly sends *javax.faces.viewstate* to maintain the current session and webpage state. After multiple redirections (steps 14 – 16), user authentication is complete, and the user can be redirected to the site where the user wanted to authenticate, which in this case was *eparaksts.lv*.

### 5.2.1 Unsuccessful authentication attempt by entering a wrong PIN 3 times

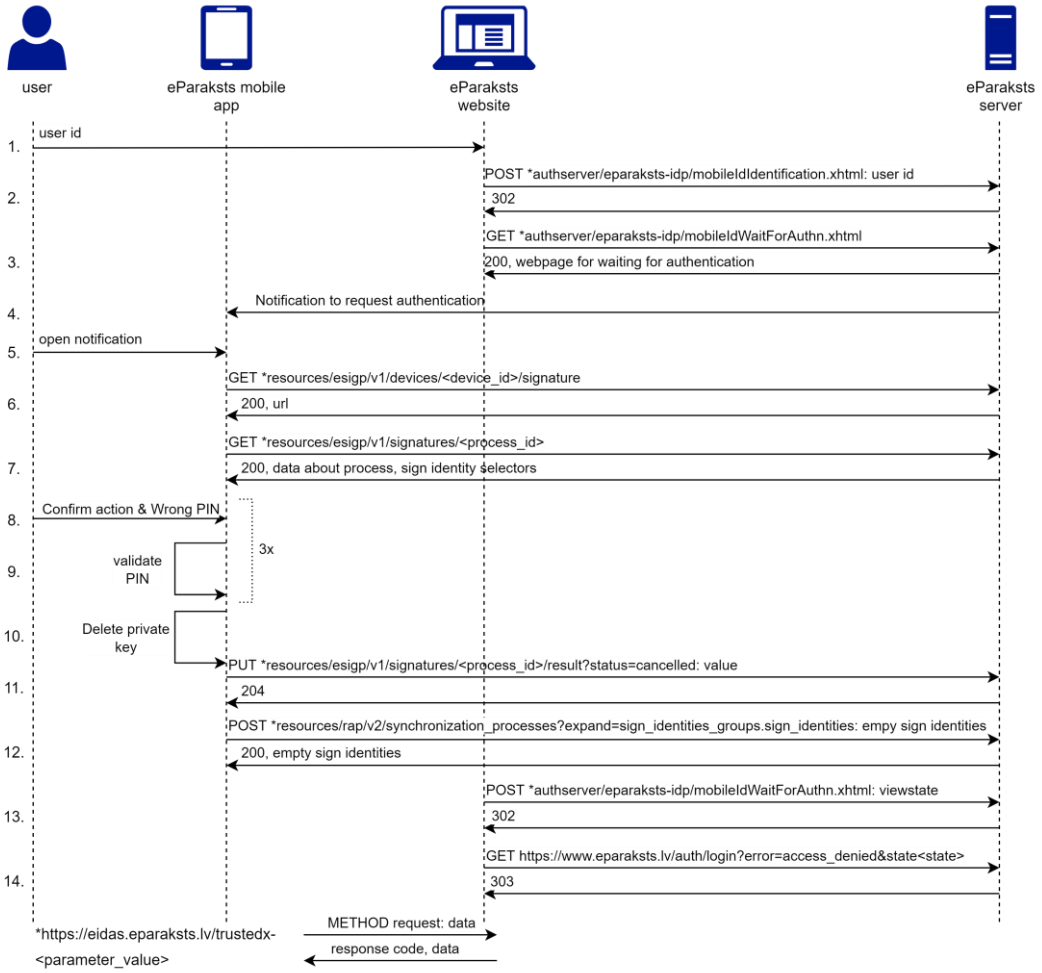


*Figure 35. A sequence diagram depicting the process of unsuccessful authentication by entering a wrong PIN 3 times*

Figure 35 depicts what happens when the user enters a PIN code incorrectly three times on the mobile app during the authentication process. Until step 8, the process is the same as in Figure 33. Then in step 8, the user enters the wrong PIN 3 times. When the PIN code validation fails for the third time (step 9), the *eParaksts mobile* app deletes the private key from the device (step 10). Then the *eParaksts mobile* app sends a request to the server informing that the authentication process has been cancelled (step 11). After that, the mobile app informs the server that the user's identity has been deleted from the device by sending a synchronisation request with the device ID and empty "*sign_identities_groups*" JSON structure (step 12). The server replies with an empty "*sign_identities_groups*" JSON structure informing that the user's authentication certificate has been annulled (step 12). The server also informs the website that the authentication request has been cancelled (steps 13 and 14).

To ensure that both the *eParaksts mobile* app and the server have the current information about the status of the user's authentication certificate, every time the mobile app is launched, the mobile app makes a synchronisation request (similar to the one in step 12) to the *eParaksts* server. That way, the mobile app learns if the server has annulled the user's authentication certificate, and the server learns if the app has deleted the user's identity from the mobile device.

### 5.2.2 Authentication on the mobile device with the eParaksts mobile app

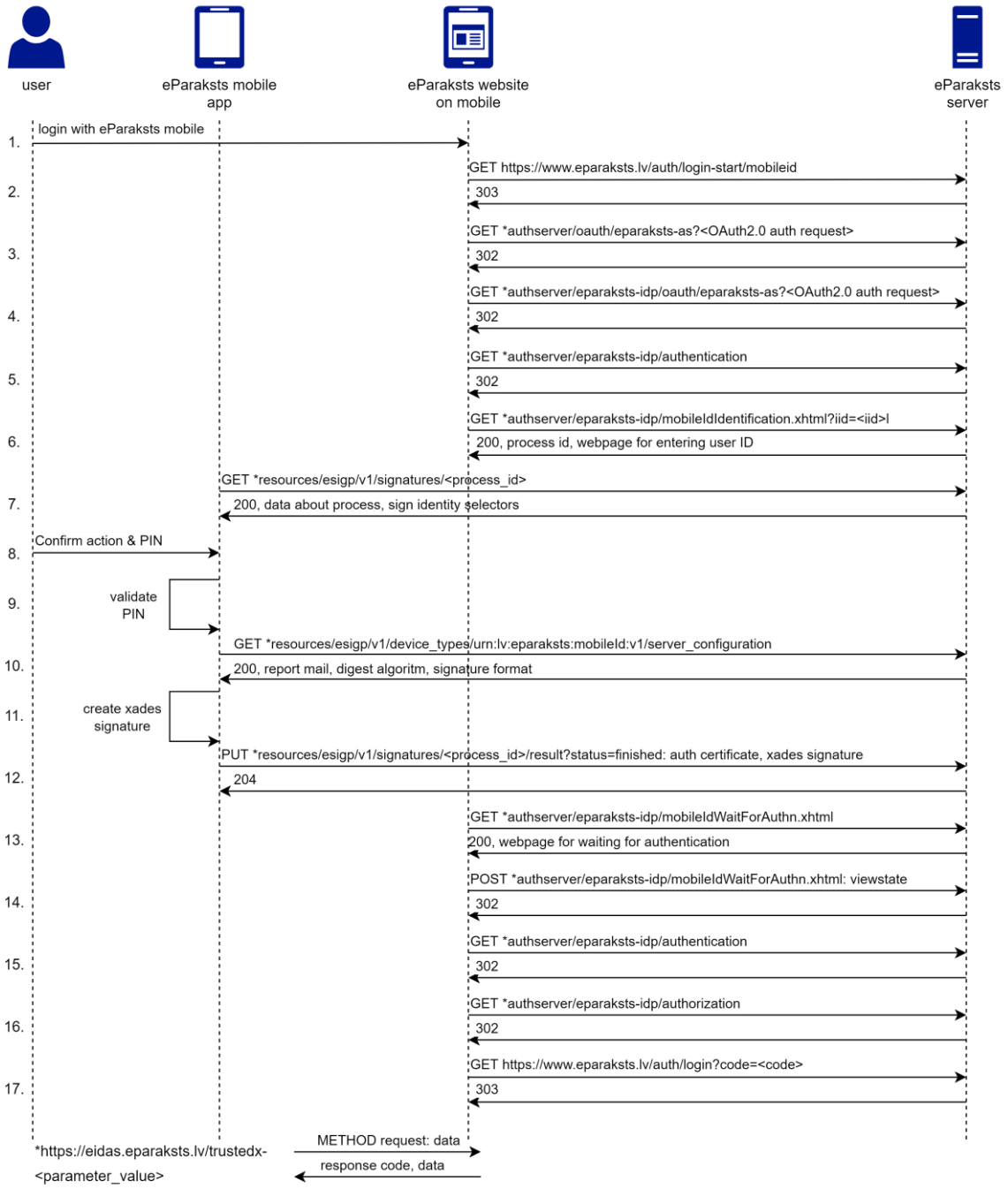


*Figure 36. A sequence diagram depicting the process of authentication with eParaksts mobile on the mobile device*

Figure 36 depicts the process of authentication with *eParaksts mobile* on the same device where the *eParaksts mobile* app is set up. The process is depicted from the point where the user chooses authentication with *eParaksts mobile* on their mobile phone's browser on the *eParaksts* website. The involved parties in this process are (1) the user with the *eParaksts mobile* app set up on their device, (2) the *eParaksts mobile* app, (3) the *eParaksts* website

(*www.eparaksts.lv*) opened in a browser on the user's mobile phone, where the user wants to be authenticated, and (4) the *eParaksts* server (*eidas.eparaksts.lv*). The *eParaksts mobile* app includes in its requests sent to the server a JWS token signed with the device authentication key in the authorisation header.

The difference between this authentication process and the authentication process described above (see Figure 33) is that the user is not asked to enter their User ID on the *eParaksts* website. Instead, the headers of the HTTP requests made by the mobile device's browser (steps 3 – 6) contain a field "*sec-ch-ua-platform*" with the value "*Android*", which indicates the platform on which the browser runs. Then, in response to the request made by the browser, the *eParaksts* server includes (see Figure 37) the process ID of the authentication request and a *JavaScript* that forces the *eParaksts mobile* app to be opened (step 6). This technique is called Mobile app deep linking [45]. The *eParaksts mobile* app is launched, and the mobile app requests information about the authentication process that corresponds to the previously received process ID (step 7). In case the user does not have the *eParaksts mobile* app installed, or it is not set up, the authentication process stops here. i.e., it is not possible to complete the authentication with eParaksts mobile, and the user is forced to use a different device or a different authentication mechanism. This behaviour is due to the fact that the *success URL* and *failure URL* are the same (see Figure 37). From step 7, the process is the same as the previously described authentication process (see Figure 33). After the user has been successfully authenticated to the *eParaksts mobile*, the mobile app is closed, and the user is redirected to the *eParaksts website*, where they wanted to authenticate.

In conclusion, the two main differences between this authentication process and the one described above (see Figure 33) are that the user does not have to provide their User ID, and no notification is sent to the *eParaksts mobile* app as the authentication process ID of the initiated authentication request is directly passed to the mobile app from the mobile device's browser.

```
<script type="text/javascript">
    //
        $(document).ready(function() {
            mobileIdBrowserCompat.init();
            if (mobileIdBrowserCompat.mustPoll()) {
                polling.start();
            }
            document.location.href = mobileIdBrowserCompat.adaptAppUri(
            "eparakstsid://citizen/signatures/kog9ld3n81a0hvtcksqpnjiupj?successurl=https%3A%2F%2F
            eidas.eparaksts.lv%2Ftrustedx-authserver%2Feparaksts-idp%2FmobileIdWaitForAuthn.xhtml&
            failureurl=https%3A%2F%2Feidas.eparaksts.lv%2Ftrustedx-authserver%2Feparaksts-idp%2Fmo
            bileIdWaitForAuthn.xhtml");
        });
        //
</script>
```

*Figure 37. JavaScript from the server's response in step 6 (see Figure 33) that is used to launch the eParaksts mobile app from the mobile browser*

## 5.3 Creating a certificate for electronic signature creation

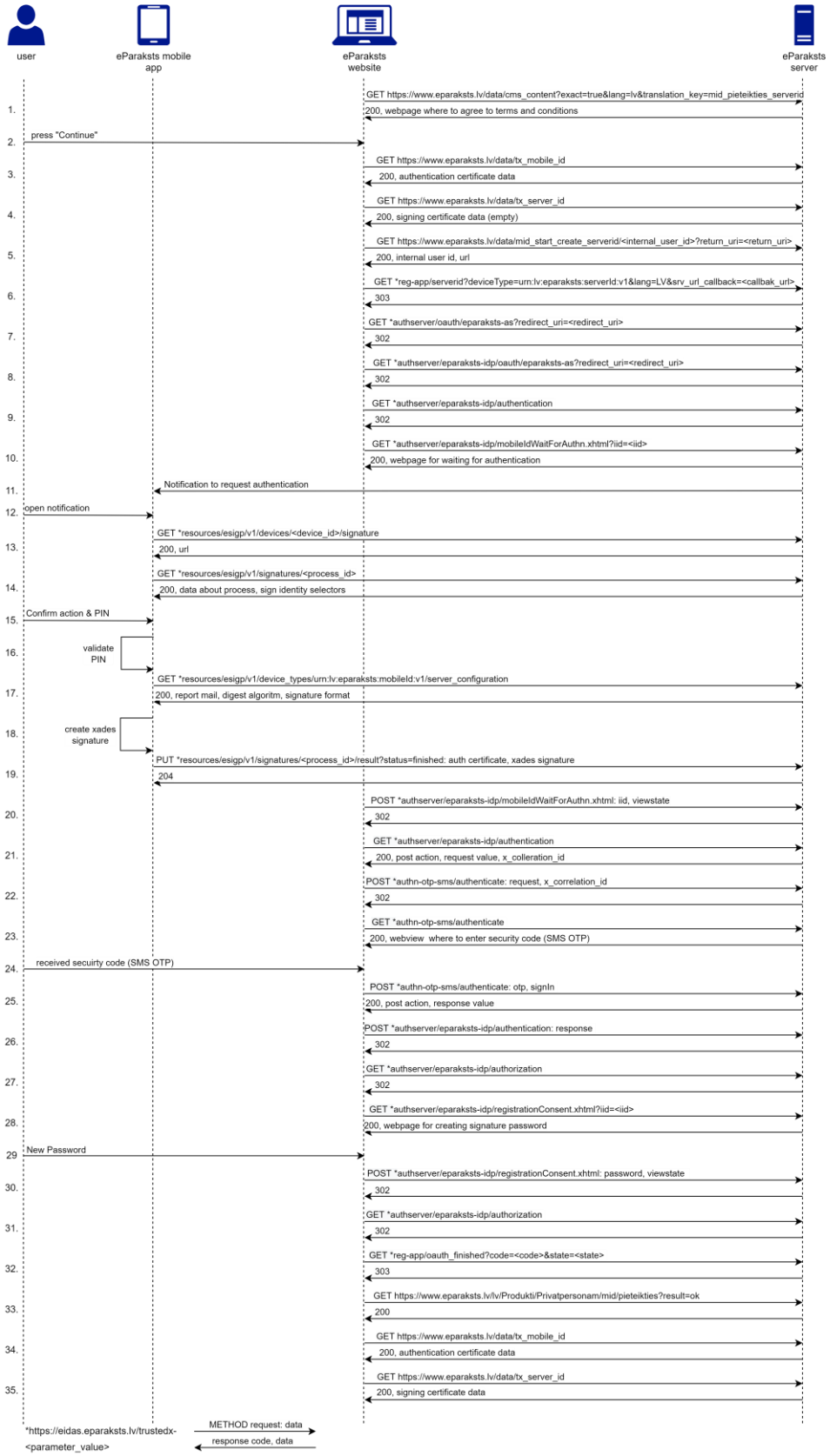


*Figure 38. A sequence diagram depicting the process of setting up the password for electronic signature creation*

Figure 38 depicts the process of setting up a password for electronic signature creation from the point where the user has initiated this process. The involved parties in this process are (1) the user with the *ePараksts mobile* app set up on their device, (2) the *ePараksts mobile* app, and (3) the *ePараksts* website (*www.eparaksts.lv*) where the user has initiated setting up the password, and (4) the *ePараksts* server (*eidas.eparaksts.lv*). This process can be split down into the following parts:

1. **Initialisation.**
    1.1. On the *ePараksts* website, the user is presented with terms related to creating the electronic signature certificate (see Figure 14) (step 1) and must agree to them by choosing to continue the process (step 2).
    1.2. The browser makes requests to the server to discover the status of the user's current certificates (steps 3 and 4). The user must have the authentication certificate and not have an electronic signature certificate. Then it starts the process of creating the user's electronic signature certificate (step 5), the server responds with a URL for completing this process, which is followed by multiple redirects (steps 6-9). Redirects lead to a webpage where authentication with the *ePараksts mobile* app is required (step 10).
2. **Authentication.**
    2.1. Since authentication with the *ePараksts mobile* app is required, steps 11 through 21 are the same as steps 4 through 14 described in Section 5.2 "Authentication with *ePараksts mobile*".
    2.2. After successful authentication with the *ePараksts mobile* app, the user must be authenticated with an additional factor. To ensure that the second-factor authentication (SMS OTP) happens on the same device where the process was initiated, the server's response contains unique values (step 21) that must be included in the subsequent request (step 22). The user is presented with a webpage for submitting an SMS OTP code (see Figure 15) (step 23). SMS is sent to the phone number that was provided during signing up for *ePараksts mobile*. The user submits the SMS OTP code (step 24), which is sent over to the server and validated to ensure that this value matches the one sent in the SMS (step 25).

**3. Creating the password and an electronic signature certificate.**

3.1. After the user has been authenticated with 2FA, it is again ensured that the next actions will take place on the same device where the user initiated the request. Hence, the server's response contains a unique (step 25) value that must be included in the subsequent request (step 26). The user is presented with a webpage for creating the password (see Figure 16) (step 28).

3.2. The user must create a password that is at least six characters long (step 29). The password is submitted to the server together with the *javax.faces.viewstate* value to maintain the session and webpage (step 30). The user's browser is redirected several times (steps 30 – 32) until certificate creation is confirmed (step 33). In this case, a 2048-bit RSA key pair is generated on the server-side. The public key of this key pair is included in the newly created electronic signature certificate. Then the browser requests data about the user's authentication certificate (step 34) and electronic signature certificate (step 35) to ensure that the process has been successfully completed.

## 5.4 Electronic signing with *eParaksts mobile*

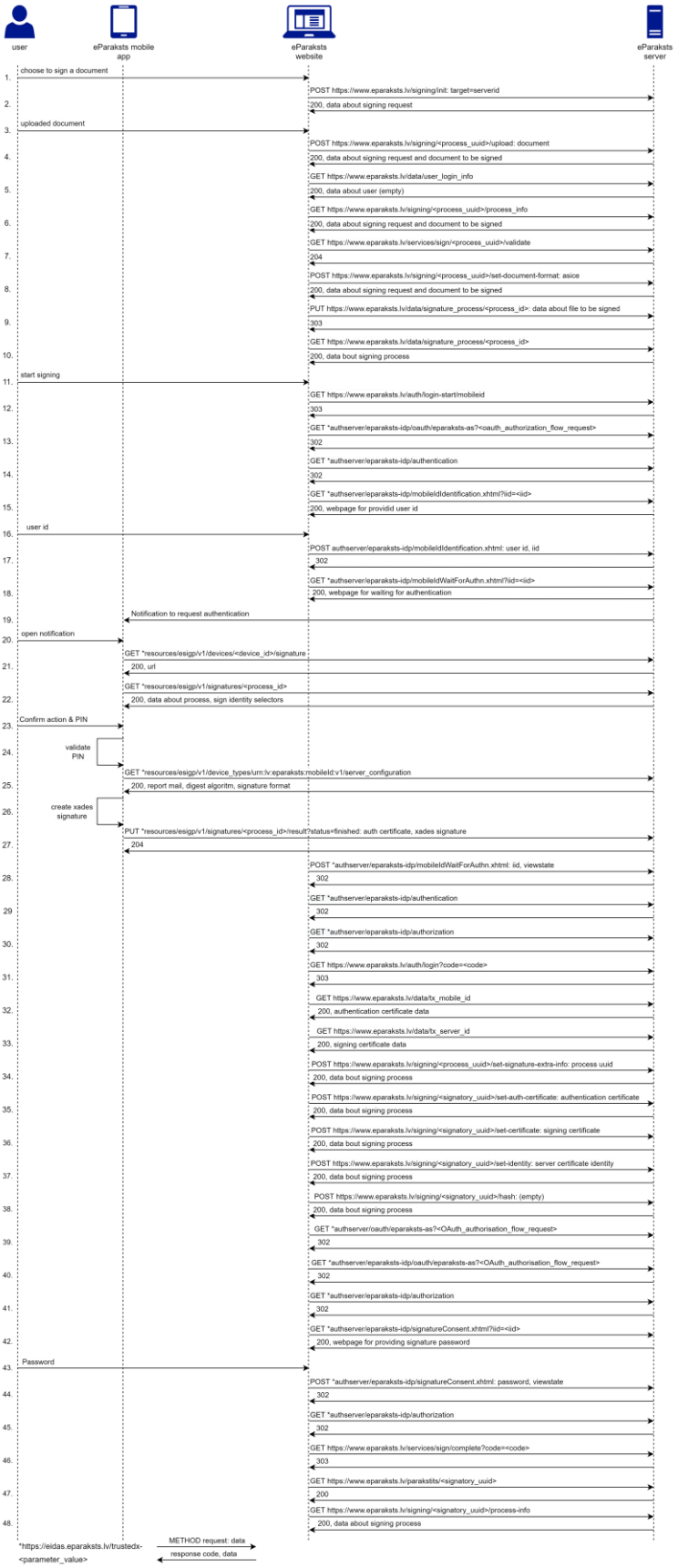


*Figure 39. A sequence diagram depicting the process of electronic signing with eParaksts mobile*

Figure 39 depicts the process of electronically signing a document using *ePараksts mobile* from the point where the user has initiated this process on the *ePараksts* website. The involved parties in this process are (1) the user with the *ePараksts mobile* app set up on their device, (2) the *ePараksts mobile* app, (3) the *ePараksts* website (*www.eparaksts.lv*) used to sign the document, and (4) the *ePараksts* server (*eidas.eparaksts.lv*).

This process can be split down into the following parts:

1. **Initialisation:**
   1.1. The user must choose the option to sign a document on the *ePараksts* website (see Figure 24) (step 1), and the server initiates the signing process by creating a signing request and assigning a unique universal identifier to the process (step 2). Then user uploads the document to the *ePараksts* website (steps 3 and 4).
   1.2. The *ePараksts* website checks whether the user has been authenticated (step 5), gets the current status of the signing process (steps 6 and 7) and sets the signature format (step 8).
   1.3. The *ePараksts* website puts received information into the signature process (steps 9 and 10).
2. **Authentication:**
   2.1. The user begins to sign the document by choosing to sign using the *ePараksts mobile* app on the website (see Figure 25) (step 11). This initiates *OAuth 2.0* authorisation code flow with several redirects (steps 12-14) that lead to the *eidas.eparaksts.lv* webpage, where authentication with the *ePараksts mobile* app is required (see Figure 17) (step 15).
   2.2. Since authentication with the *ePараksts mobile* app is required, steps 16 through 29 are the same as steps 1 through 14 described in Section 5.2 “Authentication with *ePараksts mobile*”. The user gets both authenticated and authorised to sign the document (steps 29-31).
   2.3. After that, the *ePараksts* website requests data about authentication and electronic signature certificates from the server (steps 32 and 33) to ensure that the user has these certificates. Then the website starts building a request for signing the document (step 34). The request is built on *eparaksts.lv* instead of *eidas.eparaksts.lv*

2.4. Firstly, the website sets the authentication certificate (step 35) and the electronic signature certificate (step 36) in the signing request, as well as the electronic signature certificate's identifier (step 37) and hash (step 38). Once this information is set in the signing request, the following steps can be carried out.

**3. Signing:**

3.1. For a document to be signed, it is necessary to use the signature password (that was set in Section 5.3 "Creating a certificate for electronic signature creation"). Thus, another *OAuth 2.0* authorisation code flow is initiated (steps 39-41) until a webpage for entering the password is displayed (see Figure 26) (step 42). The user submits their password (steps 43 and 44) on the *eidas.eparaksts.lv*, and the server validates the password and completes the authorisation for signing the document (steps 45-46). The server then can sign the document with the private key (whose public key is in the electronic signature certificate) and generate the signed file (depending on the user's choice, the file format can be PDF, ASiC-E or eDOC).

3.2. The signing process is finished, and the signed document becomes available to the user for download (see Figure 27) (steps 47 and 48).

# 6 The security architecture of *eParaksts mobile*

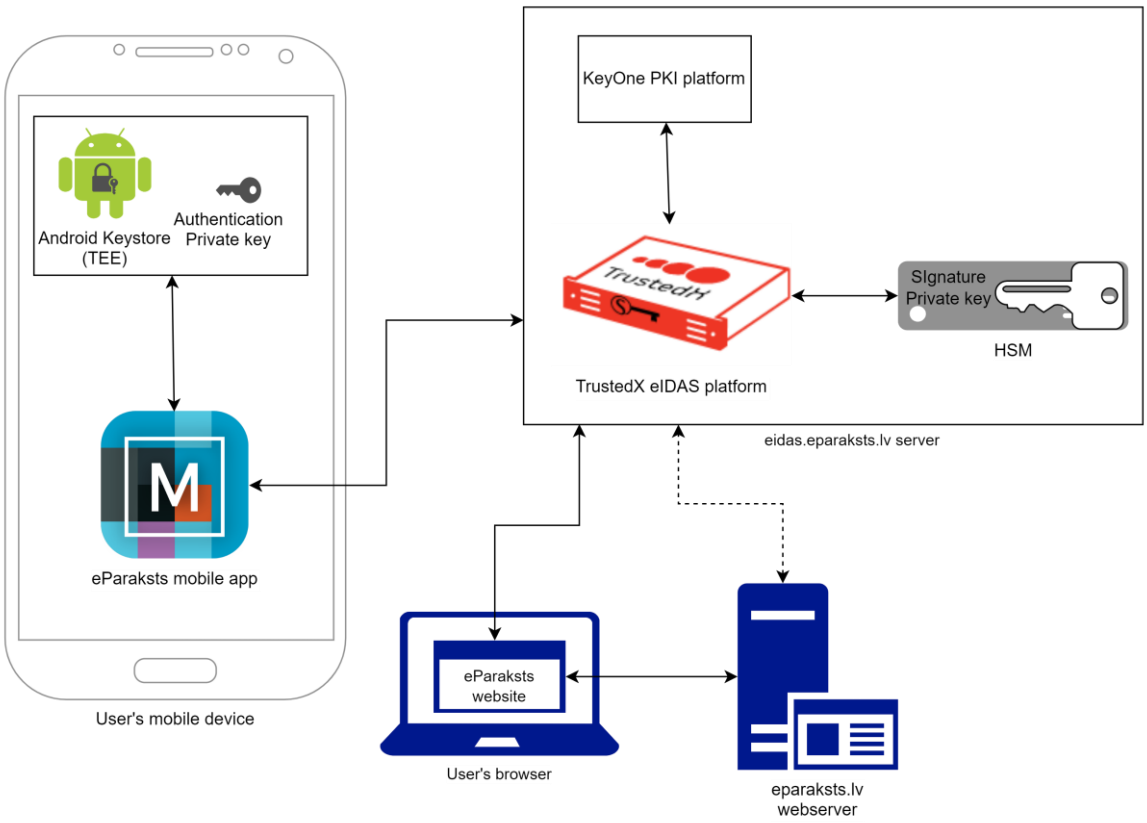


*Figure 40. The architecture of the eParaksts mobile solution (Android)*

As can be observed from network traffic analysis, *eParaksts mobile* has a hybrid architecture, meaning that it has partly device-based architecture and partly server-based architecture (see Figure 40).

The part of the solution that has a device-based architecture is authentication with *eParaksts mobile*. The key pair for the authentication certificate is generated inside Android Keystore in Trusted Execution Environment – TEE [46] (in iOS devices, TEE is called Secure Enclave – SE [47]). TEE is a secure, tamper-resistant environment in the processor [46]. The private key never leaves the TEE, and all cryptographic operations are carried on in TEE. If a mobile device does not have a TEE/SE, a software keystore is used instead [48]. All mobile devices launched with *Android* v8.0 and higher are required to have a TEE [49]. Whether an *Android* mobile device has a TEE can be verified by navigating to *Settings* → *Security & Fingerprint* → *Advanced* → *Credential Storage* and checking the value of Storage type. If it is "Hardware-backed", then the device has a TEE (the navigation steps and displayed storage type value may vary between *Android* versions). Another way to

verify if a device has a TEE, is to check (for example, on the manufacurer's website) if the device's chipset has a TEE. All *iPhones* since the *iPhone 5s* have SE [47].

The part of the solution that has a server-based architecture is electronic signing with *eParaksts mobile*, where the private key of the electronic signature certificate is kept on a Hardware Security Module (HSM) on the server-side. All cryptographic operations are carried out in HSM. The private key, in theory, can only be accessed by the person to whom the respective electronic signature certificate is issued.

As evident in both the observed network traffic and *eParaksts* General Terms and Conditions of the Trust Services [34], the *eParaksts mobile* solution is based on the *TrustedX eIDAS* platform [50], which was developed by *Safelayer Secure Communication SA* (it was acquired by Entrust Datacard at the end of 2018 [51]). The *TrustedX eIDAS* platform is an on-premise identification, authentication and electronic signature platform based on web services, and it combines authentication, single-sign-on (SSO), identity federation and trust level management functionality [52]. The *TrustedX eIDAS* platform has two additional modules that are used in the *eParaksts mobile* solution – the *KeyOne PKI* platform [34, 53] and the *Mobile ID* app [34, 54]. The *KeyOne PKI* platform is used to support the full lifecycle of PKI. The *eParaksts mobile* app is based on the *Mobile ID* app.

## 6.1 Authentication scheme

In this section, the authentication scheme that is implemented in *eParaksts mobile* is described. The authentication scheme has two main components – the creation of an authentication certificate and the authentication process.

### 6.1.1 Creation of authentication certificate

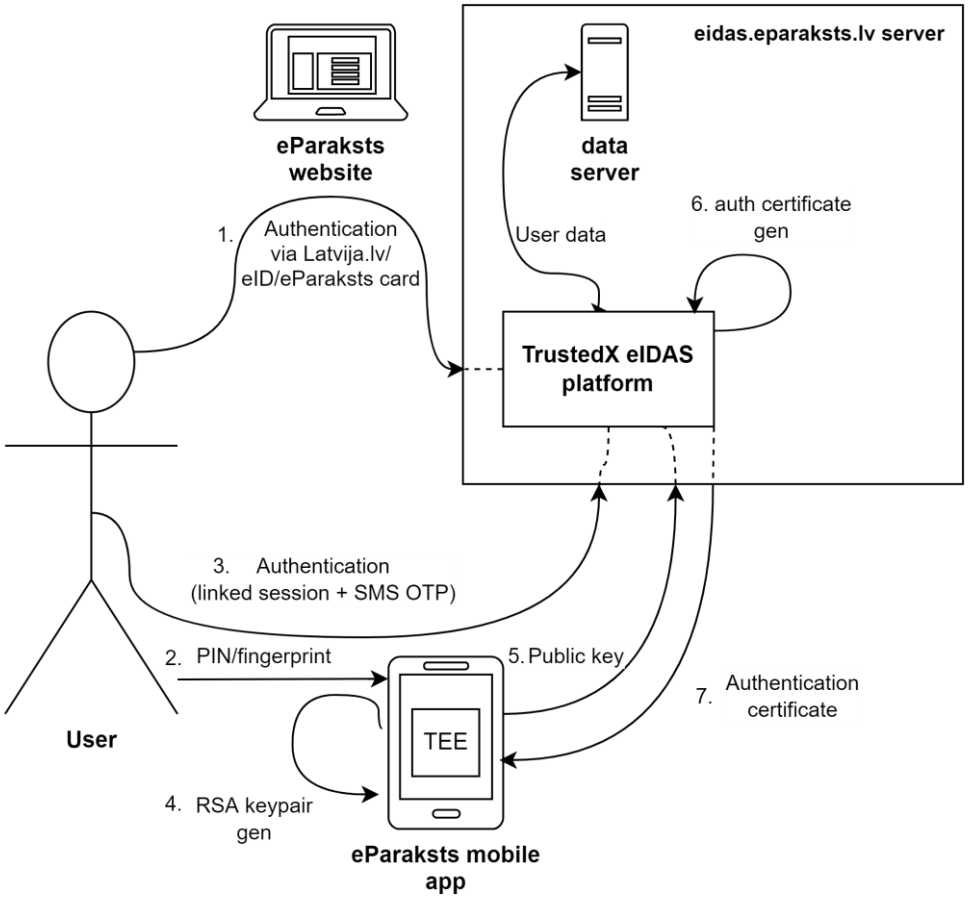


*Figure 41. Creation of authentication certificate*

Figure 41 depicts the creation of an authentication certificate in *eParaksts mobile*. Firstly, for the authentication certificate to be created, the user must be authenticated to *eparaksts.lv*. The user is authenticated against the *TrustedX* authentication server, which is part of the *TrustedX eIDAS* platform. For authentication, a user can use their national eID smart-card, existing *eParaksts* smart-card or unified authentication system "*Latvija.lv*" that supports bank authentication, which essentially means that two-factor-authentication is used. The two factors are possession (something the user has, e.g., smart-card, code calculator, bank authenticator app) and knowledge (something the user knows, e.g., the PIN code or password). Secondly, this user session on the *eParaksts* website must be linked with the *eParaksts mobile* app on the user's mobile device. Thirdly, the user must authenticate with an additional factor – submit an SMS OTP code in the *eParaksts mobile* app that is sent to the user's mobile phone number (the user provided the phone number during the registration phase where it was verified by *eParaksts*). All in all, the user is authenticated with 2FA both on the *eParaksts* website and the *eParaksts mobile* app. For the mobile app, the

authentication factors are the possession (the SIM card linked to the user's phone number) and knowledge. In this case, the knowledge factor comes from the user's previously established session with the website because the two authentication factors there were possession and knowledge.

After the user has been authenticated on the mobile app, the 2048-bit RSA key pair is generated on the user's mobile device in the keystore [48]. Then the *eParaksts mobile* app sends the public key to the *TrustedX eIDAS* platform, where the certificate is generated based on the user's identity acquired during authentication and sent over to the *eParaksts mobile* app. Then the certificate can be installed on the device. The private key is protected by a PIN code (or optionally biometric credentials) that was created at the beginning of the *eParaksts mobile* app setup. When the user enters their PIN (or provides their biometric credentials), they authorise the private key usage in TEE/SE [55]. The authentication certificate contains a critical *x509v3* key usage property "Digital signature", which is commonly used for entity authentication [56]. It also contains the *x509v3* extended key usage property "TLS Web Client Authentication".

On the *TrustedX eIDAS* platform, a "sign identity" is created, which links together the user's identity, mobile device's identifiers and the created authentication certificate. This structure allows to uniquely identify a user using a particular device on which the private key related to the authentication certificate is saved. The authentication certificate is linked to the user's mobile device's unique identifier (see Figure 42). The "sign identity" structure does not include information about which keystore (hardware or software) was used for the private key nor whether the user enabled biometric credential usage or not. This information is never shared with the *eidas.eparaksts.lv* server at any point in this process. However, the keystore type is accessible to the app itself (see Appendix II "Code snippets from the source code of the *eParaksts mobile* app").

The authentication private key is generated and stored on the keystore. However, it looks like the PIN code is verified outside the keystore because, during PIN code verification, methods related to the keystore are not used (see Appendix II "Code snippets from the source code of the *eParaksts mobile* app"). There is an option to protect the private key on the Android keystore with a subset of the user's lock screen credentials [55], but this has not been implemented in the *eParaksts mobile* app. In that case, the user would use the lock screen credentials to authorise key usage, and the credentials would be verified in the keystore.

```
{
    "sign_identities_groups": [
        {
            "access": [
                {
                    "user_id": "e5e7af8eb6459a512e9428005d79b130"
                }
            ],
            "device": {
                "id":
"be4a620d46cfe6bab839b84ad147a991f8f83523622fbe4b92b9b488380db0a7",
                "url": "https://eidas.eparaksts.lv/trustedx-
resources/esigp/v1/devices/be4a620d46cfe6bab839b84ad147a991f8f83523622fbe4b92b9b
488380db0a7"
            },
            "domain": "citizen",
            "expiration": "2025-03-14T13:35:38Z",
            "id": "vr73rpe5j7aeut7u0d783v5b9e412acr",
            "labels": [
                "mobileidVersion1",
                "eparaksts",
                "mobileid"
            ],
            "self": "https://eidas.eparaksts.lv/trustedx-
resources/rap/v2/sign_identities_groups/vr73rpe5j7aeut7u0d783v5b9e412acr",
            "sign_identities": [
                {
                    "access": [
                        {
                            "user_id": "e5e7af8eb6459a512e9428005d79b130"
                        }
                    ],
                    "details": {
                        "certificate": "<Base64 encoded certificate>",
                        "public_key": "<Base 64 encoded public key of the
certificate>"
                    },
                    "device_id":
"be4a620d46cfe6bab839b84ad147a991f8f83523622fbe4b92b9b488380db0a7",
                    "domain": "citizen",
                    "id": "78v1vvhaq3svpiqelsraoe44mi",
                    "labels": [
                        "mobileidVersion1",
                        "eparaksts",
                        "mobileid",
                        "x509:keyUsage:digitalSignature"
                    ],
                    "self": "https://eidas.eparaksts.lv/trustedx-
resources/esigp/v1/sign_identities/78v1vvhaq3svpiqelsraoe44mi",
                    "status": {
                        "value": "enabled"
                    },
                    "type": "pki:x509"
                }
            ],
            "sign_identities_schemes_group": {
                "id":
"urn:eidas:eids:scheme:eparaksts:mobile&eparaksts:mobileid",
                "url": "https://eidas.eparaksts.lv/trustedx-
resources/rap/v2/sign_identities_schemes_groups/urn:eidas:eids:scheme:eparaksts:
mobile&eparaksts:mobileid"
            }
        }
    ]
}
```


*Figure 42. JSON structure representing "sign identity" (from server's response in step 28 in Figure 32)*

### 6.1.2 Authentication

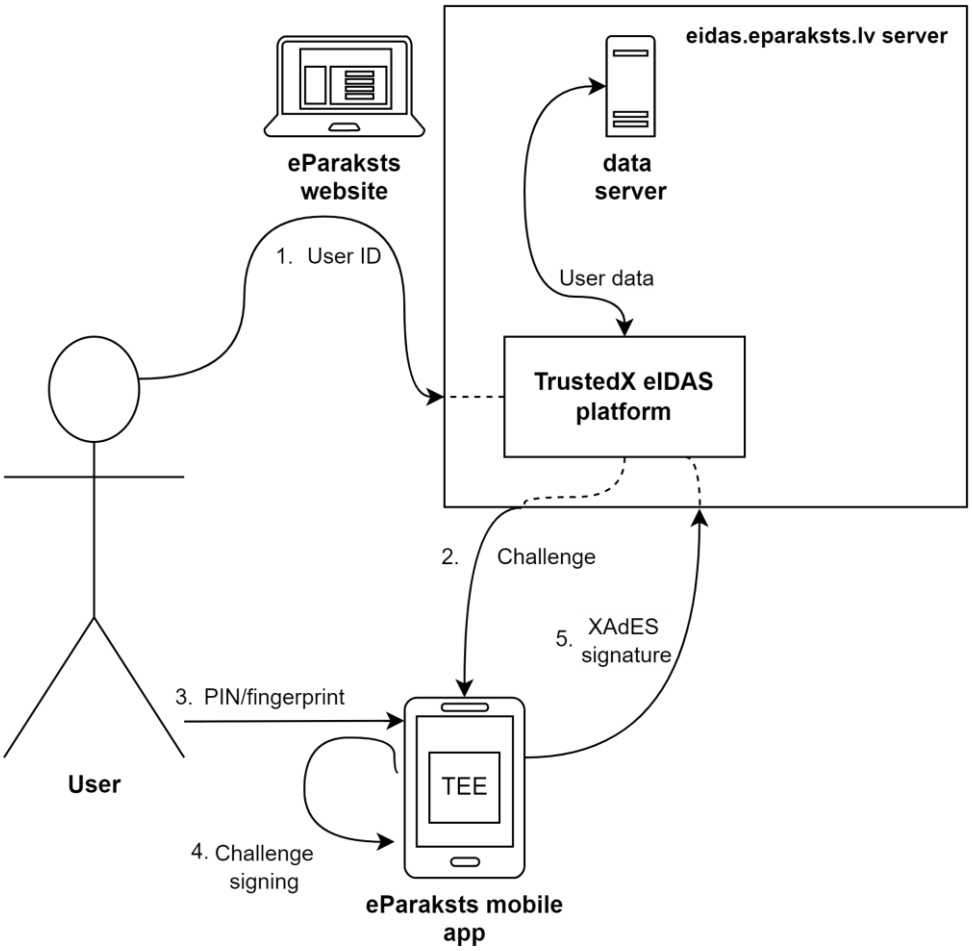


*Figure 43. Authentication scheme*

Figure 43 depicts user authentication to the *eParaksts* website using *eParaksts mobile* authentication. The authentication scheme implemented in *eParaksts mobile* is a challenge-response authentication scheme based on PKI. With the *eParaksts mobile* app, 2FA is achieved – the first factor is possession (mobile device/app), and the second factor is either knowledge (the PIN code) or inherence (the user's biometric credentials).

The authentication process begins when the user selects *eParaksts mobile* as the authentication method on the website and submits their User ID to the *eidas.eparaksts.lv* website, which is sent to the *TrustedX* authentication server. The server sends an authentication notification to the user's *eParaksts mobile* app using some cloud messaging service (in this case, Firebase Cloud Messaging (FCM)), which was configured during the *eParaksts mobile* app setup phase. When the user clicks on the notification, the *eParaksts mobile* app submits a request to the server to see whether there are any authentication

requests pending. If there are, then the *eParaksts mobile* app receives information about this specific authentication request, including the challenge that must be signed. The name of the service requesting authentication (in this case, the service is *eParaksts*) is displayed to the user. The user then must explicitly confirm this authentication action by clicking a "Confirm" button in the app.

The user must enter the PIN code (or biometric credentials) that protects the private key. The *eParaksts mobile* app locally verifies the PIN code. If the PIN code is entered incorrectly three times, the mobile app would delete the private key and send a request to the server to annul the authentication certificate. When the PIN is successfully verified, the challenge is then signed with the private key in the TEE (or software keystore if TEE is not available). Then the *eParaksts mobile* app generates an XAdES signature that is sent to the *TrustedX* platform, where the response is verified, and upon successful verification, the user is authenticated.

An example of an XAdES signature can be seen in Appendix III "Example of XAdES signature". The main information that is signed is the specific authentication process identifier and the identifier of the service requesting authentication. This XAdES signature does not include any information regarding whether the challenge was signed inside the software or hardware keystore, nor if biometric credentials were used to authorise the private key usage.

According to the LVRTC integration guidelines [57], *OAuth 2.0* authorisation code flow is used for letting other e-service providers offer authentication with *eParaksts mobile*. The e-service provider redirects the user to the eParaksts mobile website for authentication. After a successful authentication on *eParaksts mobile,* and after the user has given explicit consent, the e-service provider receives an authorisation code. Then via back-channel, this authorisation code is exchanged for an access token. The access token, in turn, is used to get information about the user, such as legal name and surname and national identification code. It is also possible to request the "sign identity" structure via back-channel to get information about the user's authentication and electronic signature certificates.

## 6.2 Electronic signing scheme

In this section, the electronic scheme that is implemented in *eParaksts mobile* is described. The electronic signing scheme has two main components – the creation of an electronic signature certificate and the electronic signing process.

### 6.2.1 Creation of electronic signature certificate

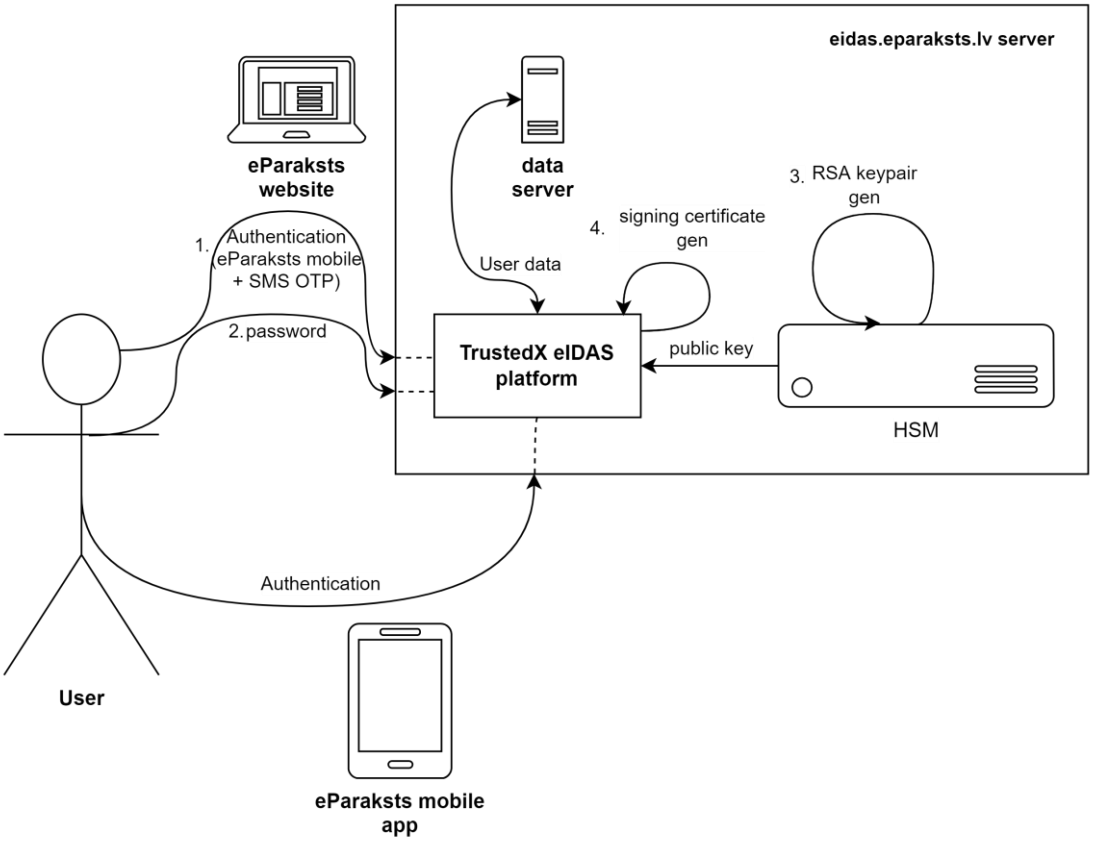


*Figure 44. Creation of an electronic signature certificate*

Figure 44 depicts the process of creating an electronic signature certificate. For an electronic signature certificate to be created, the user must be authenticated using 2FA. Firstly, with the *eParaksts mobile* app to the *eParaksts* website. Secondly, the user must submit an SMS OTP code to the *eParaksts* website that is sent to the user's mobile phone number (similar to the authentication during the creation of the authentication certificate). Once the user is fully authenticated on the *eParaksts* website, the user must create a 6-character password that will authorise the usage of the private key related to the electronic signature certificate. When the user enters their password, they authorise the private key usage in the HSM. Once the password is submitted to the *TrustedX eIDAS* platform via the *eParaksts* website, the 2048-bit RSA key pair is generated on HSM. The public key of this key pair is then included in the newly generated electronic signature certificate. The electronic signature certificate contains a critical *x509v3* key usage property "Non-Repudiation". This property denotes that the key can be used to create legally binding electronic certificates [56]. In theory, the key on HSM can only be accessed with the user's 6-character password, and this certificate is linked with the user's identity in the *TrustedX eIDAS* Platform [48].

### 6.2.2 Electronic signing

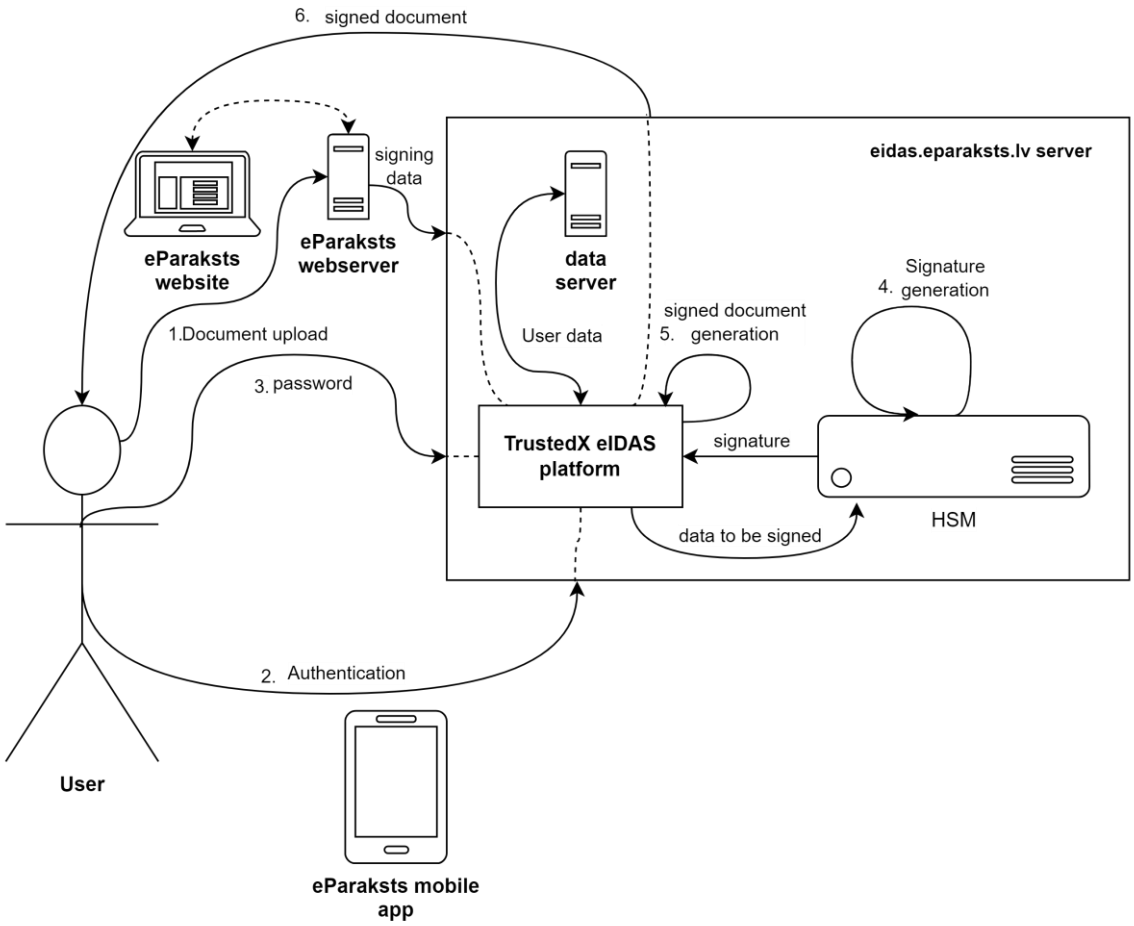


*Figure 45. Electronic signing scheme*

Figure 45 depicts the process of electronically signing a document with *eParaksts mobile*. The user must upload the document to be signed to the *eParaksts* website and select the desired format (PDF, eDOC or ASiC-E) of the signed file. On the back-end, data to be signed is shared with the *TrustedX eIDAS* platform. Then the user must authenticate to the *TrustedX* authentication server with the *eParaksts mobile* app (the authentication scheme is carried out). However, the *eParaksts mobile* app does not differentiate between just an authentication process and authentication for signature creation, as all XAdES signatures created during the authentication process include the same information. Next, the user must submit their signature password via the *eParaksts* website to the *TrustedX eIDAS* platform. Then the *TrustedX eIDAS* platform sends data to be signed to the HSM, where the signature is generated with the user's private key related to the electronic signature certificate. The HSM sends back the signature, and the *TrustedX eIDAS* platform can generate the signed

document. Afterwards, the signed document is available for download in the selected format from the *eParaksts* website.

According to the LVRTC integration guidelines [57], other e-service providers can integrate with *eParaksts mobile* to offer electronic signing functionality on their websites. For example, this functionality has been implemented in an e-service *e-address* [58], where users can send electronically signed messages to government institutions. Electronic signing functionality is implemented with *OAuth 2.0* authorisation code flow similarly to how it is implemented on the *eParaksts* website. The users are redirected to the *eidas.eparaksts.lv* website for authentication with *eParaksts mobile* and for authorising the signature creation with the signature password. The data to be signed is sent to the *eIDAS TrustedX* platform via back-channel, where the data is signed, and the signature is returned in response to this request.

# 7 *eParaksts* and eIDAS

This chapter covers relevant legal aspects of *eParaksts mobile* authentication and electronic signing solutions regarding eIDAS regulation.

## 7.1 *eParaksts mobile* eID identification scheme

eID identification schemes are covered in eIDAS regulation [59]. However, it is not mandatory for an eID scheme to comply with the regulation. The EU Member States can voluntarily notify the European Commission of their eID schemes. If an eID scheme gets notified, it means that it should be recognised in the other Member States for cross-border authentication. In eIDAS, there are defined three levels of assurance (LoA) for eID schemes – low, substantial and high [59]. LoA refers to a degree of confidence in a person's claimed or asserted identity.

Latvia notified the European Commission of its eID schemes on the 15th of November, 2018. On the 18th of December 2019, European Commission included these eID schemes in the list of notified eID schemes. Among Latvia's notified eID schemes, there is also the *eParaksts* eID scheme which includes the authentication with the *eParaksts mobile* app [60]. Based on the Opinion of the Cooperation Network [61], the *eParaksts* eID scheme has two levels of assurance – substantial and high. If the private keys are stored in TEE/SE, then the LoA is high. Otherwise, if they are stored in the software keystore, LoA is substantial. Moreover, if biometric identification is enabled, then LoA is also substantial. As observed from the analysed network traffic, the *eParaksts* server is not informed whether the private key is stored in the hardware or software keystore and whether biometric identification is enabled or not. Therefore, the *eParaksts* server can only guarantee a substantial level of assurance.

## 7.2 *eParaksts mobile* electronic signature

As per eIDAS regulation [59], for an electronic signature to be recognised as a qualified electronic signature (QES), meaning that it would have the same legal strength as a handwritten signature, it must meet the requirements for an advanced electronic signature, be based on a qualified certificate for electronic signatures and be created with a qualified electronic signature device (QSCD). That means that for a signature that has been created *with eParaksts mobile* to be recognised as QES, *eParaksts mobile* should have a status of QSCD.

To receive a status of a QSCD, the device must comply with requirements laid out in eIDAS regulation, specifically in Annex II [59]. To ensure that such a signing device complies with the security requirements for a QSCD, it must receive at least Common Criteria Evaluation Assurance Level 4+ (CC EAL 4+) [16].

In 2010, the *TrustedX eIDAS* platform received CC EAL 4+ [62] (which now is considered archived [63]), and in 2014 *KeyOne* also received CC EAL 4+ [53] (which is still current). Moreover, the *TrustedX eIDAS* platform has a remote QSCD status (the current certificate was issued on the 29th of October 2018) with a note that it should be managed by a QTSP [64].

On the 26th of June 2020, the Supervisory Committee of Digital Security (the national supervisory body of Latvia) recognised *eParaksts* as a qualified trust service that is managed by LVRTC – a qualified trust service provider (QTSP) [65]. Subsequently, on the 28th of June 2020, *eParaksts mobile* gained its ability to issue qualified electronic signatures (QES) [3]. It means that signatures created with *eParaksts mobile* have the same legal strength in the EU as handwritten signatures have.

On the 14th of June 2021, certification body *QSCert* issued a certificate [66] to LVRTC that states that LVRTC provides "qualified trust service for creation and validation of qualified certificates for electronic signature", this certificate confirms compliance with eIDAS regulation (including the beforementioned Annex II) [59] and refers to relevant trust service policies of *eParaksts* solution (in this case the relevant policy is "*eParaksts*" Trust service policy v2.2 [48] (the current version is v2.5).

# 8 Security implications

In this chapter, potential security implications emerging from the architecture of the *eParaksts mobile* authentication and electronic signing schemes are discussed.

## 8.1 Authentication with *eParaksts mobile*

In the *eParaksts mobile* authentication scheme, the user's private key related to the authentication certificate is kept on the user's device in the hardware keystore (or in the software keystore if the hardware keystore is not available). Therefore, the server-side cannot generate a valid signature for the authentication challenge, even if it is compromised. Furthermore, the security of the authentication scheme depends on the security of the mobile device and the security of the private key.

Currently, it is possible to use the *eParaksts mobile* app with a software keystore that is considerably less secure than a hardware keystore [67]. An attacker with full control over the user's mobile device could get root access to the device and extract the private key. If the option to use the software keystore were disabled in the *eParaksts mobile* app, then risks associated with using software keystores would be mitigated, and one of the factors lowering the eID scheme's LoA would be removed as well. While *eParaksts* is moving towards this goal by requiring *Android 8.0* or higher with their newest *eParaksts mobile* app version, it is still possible to download and use an older app version because the *eParaksts* server does not have any restrictions regarding mobile app versions. Moreover, the *eParaksts* server does not receive information about which keystore is used for the private key. That means that the server does not have the means to validate the level of assurance and, therefore, can guarantee only a substantial LoA.

However, even if the TEE is used for storing the private key, several studies [67-70] have been done about vulnerabilities in TEE that allow compromising stored credentials. While the *eParaksts mobile* app cannot treat these vulnerabilities, it would be worthwhile to test its behaviour using a compromised TEE to discover possible weak points in the app. Additionally, it is likely that gaining just the private key is not sufficient to successfully authenticate on the user's behalf because the app's HTTP requests always contain a JWS token signed with the device's authentication key in the authorisation header. Hence, the attacker would also need to get access to that authentication key. Moreover, the use of TEE protects the private key from being used and extracted outside the mobile app's processes.

However, it does not protect against using the private key via the mobile app if the mobile app is compromised [55].

Besides, the PIN code is not verified inside the keystore. It means that the PIN code could be brute-forced or extracted from the mobile app, regardless of which keystore is used. This would enable an attacker to use the private key and authenticate on the legitimate user's behalf. However, to execute such attacks, the attacker would need to have physical or logical (root-level) control over the user's mobile device.

Additionally, it should be analysed whether any clone detection controls are implemented to prevent the usage of a cloned instance. If a software keystore is used, an attacker can theoretically clone the app instance into another device and use it on behalf of the legitimate user.

In the "*eParaksts*" Trust service policy [48], it is stated that the user is responsible for keeping the PIN code and mobile device under their control, which raises questions about the security of a treat model where the user's device is under adversary's control.

Furthermore, an attacker might not even need access to the user's app/mobile device to take advantage of the *eParaksts mobile* authentication. Suppose an adversary learns the user's User ID by out-of-band means. In that case, they could initiate multiple requests until the user authenticates, considering that the notification "*eParaksts* requests for identification" is rather vague, and the user does not have a reason to doubt the legitimacy of the notification. Moreover, if the attacker tries to authenticate with the legitimate user's User ID at the same time as the user, they might trick the user into approving the malicious authentication request. To address this attack vector, on the *eParaksts* FAQ page [71], there are instructions suggesting that the user should cancel any authentication requests that they did not initiate themselves. However, a more effective solution would be to introduce more information in the authentication request and display an OTP code in the authentication request and authentication form on the website. Then the user could verify that the authentication request on the mobile app corresponds with the initiated authentication on the website (similarly as it is done in the *Smart-ID* solution).

The associated risk of an adversary deleting an existing authentication certificate and creating a new one is relatively low because the user would receive an email and could take necessary actions (e.g., annul the adversary's certificate). Additionally, the attacker would require at least temporary control over the user's SIM card because SMS OTP codes are

used as an additional authentication factor. Moreover, it is not possible for an adversary to change a user's email address or phone number because this information can only be changed by annulling the existing contract with LVRTC [72].

## 8.2 Signature creation with *eParaksts mobile*

In the *eParaksts mobile* electronic signing scheme, the user's private key related to the electronic signature certificate is kept on the HSM on the server-side. The user's signature password is necessary to authorise the usage of this private key, and the user must be authenticated with the *eParaksts mobile* app to sign a document. The security of the electronic signing scheme relies not only on the security of the private key on the HSM but also on the security of the *eParaksts mobile* app.

According to the QSCD certification report [73], the password for authorising the usage of the signature private key is only shared between the HSM and the user, which means that the password is verified inside the HSM. It is also stated that a request to create a signature cannot be made directly through the QSCD but only through a server signing application. Moreover, the user should be authenticated with at least a substantial level of assurance (which is in line with the *eParaksts mobile* authentication scheme). However, how the user's identity is provided is outside the scope of the aforementioned certification report. The observed network traffic did not provide any indications that the authentication is linked to the signature creation, i.e., the authentication challenge does not include any information about the signature creation process.

An attacker with direct access to the *eIDAS TrustedX* platform could potentially bypass the authentication requirement and create an electronic signature without the legitimate user's knowledge. To do that, the attacker would have to find out the user's password, and there are several ways that could be achieved. Firstly, someone with direct access to the *eParaksts* server could easily sniff the password submitted by the legitimate user. Secondly, if the password retry count is not implemented on the HSM, the attacker could try to brute-force the password. Thirdly, the signature creation certificate is not annulled after retry attempts have been exhausted. Therefore, if the corresponding private key is not deleted, it still might be possible to issue signatures with such a certificate. In that case, the retry count limit does not provide any obstacles to the attacker. Ultimately, the certificate should be annulled after these 15 unsuccessful password entering attempts because the user is still required to annul

the certificate manually in order to regain the ability to create electronic signatures, as it is not possible to keep using the existing certificate from the client-side.

It is necessary to do comprehensive research about the server-side implementation of the signature creation with *eParaksts mobile*. However, researching these issues is not possible without access to the server-side setup of the *eParaksts mobile* solution.

Similarly to the authentication scheme, the associated risk of an adversary deleting an existing electronic signature certificate and creating a new one is relatively low because the user would receive an email and could take necessary actions (e.g., annul the adversary's certificate). Moreover, SMS OTP code is used as an additional authentication factor which would require at least temporary control over the user's mobile device to create a new certificate.

# 9 Conclusions

The goal of this thesis was to understand and describe the authentication and electronic signing schemes of Latvian eID solution *eParaksts mobile*, as well as understand the architecture of the *eParaksts mobile* solution.

To achieve this goal, the network traffic between the *eParaksts mobile* client (mobile app and browser) and the server was analysed. To obtain relevant network traffic, a MitM attack was executed. Moreover, the *eParaksts mobile* app was modified to make the MitM attack possible.

The research performed in this work allowed us to make the following conclusions. The *eParaksts mobile* solution is based on the *TrustedX eIDAS* platform and the *Mobile ID* app developed by *Safelayer Communications SA.*

Based on where the private keys are located, the architecture of *eParaksts mobile* is hybrid. The private key necessary for authentication is kept on the device in TEE/SE (or software keystore if TEE/SE is not available). Hence, from this perspective, the solution has a device-based architecture. The implemented authentication scheme is a challenge-response scheme that is based on PKI. When authentication with *eParaksts mobile* is initiated, the *eParaksts* server creates a challenge that must be signed with the user's authentication private key. The user can authorise the usage of their private key with a 4-digit PIN or biometric credentials. The PIN code is validated locally on the device. Authentication with *eParaksts mobile* is two-factor authentication – possession of the device/*eParaksts mobile* app is one factor, and knowledge of the PIN code or inherence of the biometric credentials is the second factor.

However, the private key necessary for electronic signatures is kept on an HSM on the server-side, making the architecture server-based. To sign a document with *eParaksts mobile*, the user must authenticate themselves with the *eParaksts mobile* app and authorise the usage of their private key with a 6-character password. The password is validated on the server-side. The usage of the *eParaksts mobile* app ensures that two-factor authentication is performed before issuing the signature.

The *eParaksts* eID authentication scheme is notified under eIDAS regulation and has two levels of assurance – high and substantial. However, the *eParaksts* server does not receive information from the *eParaksts mobile* app about the conditions that impact the assurance

level –keystore type and if authentication with biometric credentials is enabled. Therefore, only a substantial level of assurance can be assumed.

The *eParaksts mobile* electronic signature scheme supports the creation of QES, which means that within the EU, these signatures have the same legal strength as handwritten signatures. Signatures created with *eParaksts mobile* are QES because the *TrustedX eIDAS* platform has been recognised as a remote QSCD, and LVRTC is recognised as QTSP.

To conclude, this thesis provides a foundation for future work related to the security analysis of the *eParaksts mobile* solution. The integration between the *eParaksts mobile* app and device keystore should be studied more to identify potential vulnerabilities with the private key management, for example, how easy it would be to bypass PIN validation. Additionally, private key management on the server-side could also be researched to understand the level of control or lack thereof the user has over their private key stored on the server-side HSM. Particularly how easy it is to bypass the authentication with *eParaksts mobile* and how hard it would be for someone with direct access to the QSCD to gain access to the user's electronic signature private key and create signatures without their knowledge.

## References


[1] Vineta Sprugaine, "eParaksts in Latvia for 10 years (in Latvian)," 04.10.2016. Available: https://lvportals.lv/dienaskartiba/282229-eparakstam-latvija-jau-10-gadi-2016

[2] eParaksts, "Submit application to receive eSignature mobile," 11.12.2017 Available: https://www.eparaksts.lv/en/about_us/News/Submit%20application%20to%20receive%20eSignature%20mobile

[3] eParaksts, "eParaksts mobile signature has also received the highest qualification in European Uninon (in Latvian)," 22.07.2020 Available: https://www.eparaksts.lv/lv/par_mums/Jaunumi/eParaksts_mobile_paraksts_sanemis_augstako_kvalifikaciju_ari_Eiropas_Savieniba

[4] LA.LV, "eParaksts mobile in Latvia for 15 years. It has been used by a fifth of the country's population aged 15 and over. (in Latvian)," 04.10.2021. Available: https://www.la.lv/eparakstam-latvija-jau-15-gadi

[5] Siddhartha Arora, "National e-ID card schemes: A European overview," *Information Security Technical Report*, vol. 13, no. 2, pp. 46-53, 2008. Available: https://doi.org/10.1016/j.istr.2008.08.002.

[6] Michael Kubach *et al.*, "SSEDIC. 2020 on Mobile eID," *Open Identity Summit 2015*, 2015. Available: https://dl.gi.de/handle/20.500.12116/1971.

[7] Antonio Ruiz-Martínez *et al.*, "A survey of electronic signature solutions in mobile devices," *Journal of Theoretical and Applied Electronic Commerce Research*, vol. 2, no. 3, pp. 94-109, 2007. Available: https://doi.org/10.3390/jtaer2030024.

[8] Thomas Zefferer, "A Sustainable Architecture for Secure and Usable Mobile Signature Solutions," Cham, 2016, pp. 343-364: Springer International Publishing. Available: https://doi.org/10.1007/978-3-319-30996-5_17.

[9] Christof Rath *et al.*, "Design and application of a secure and flexible server-based mobile eID and e-Signature solution," *International Journal on Advances in Security*, vol. 7, no. 3&4, pp. 50-61, 2014. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.680.424&rep=rep1&type=pdf#page=22.

[10] Thomas Zefferer, "A server-based signature solution for mobile devices," in *Proceedings of the 12th International Conference on Advances in Mobile Computing and Multimedia*, 2014, pp. 175-184. Available: https://doi.org/10.1145/2684103.2684142.

[11] Vanga Odelu, Ashok Kumar Das, and Adrijit Goswami, "A secure biometrics-based multi-server authentication protocol using smart cards," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 9, pp. 1953-1966, 2015. Available: https://doi.org/10.1109/TIFS.2015.2439964.

[12] Min Zhang, Jiashu Zhang, and Wenrong Tan, "Remote three-factor authentication protocol with strong robustness for multi-server environment," *China Communications*, vol. 14, no. 6, pp. 126-136, 2017. Available: https://doi.org/10.1109/CC.2017.7961369.

[13] Clemens Orthacker, Martin Centner, and Christian Kittl, "Qualified mobile server signature," in *IFIP International Information Security Conference*, 2010, pp. 103-111: Springer. Available: https://doi.org/10.1007/978-3-642-15257-3_10.

[14] Christof Rath *et al.*, "Encryption-based second authentication factor solutions for qualified server-side signature creation," in *International Conference on Electronic Government and the Information Systems Perspective*, 2015, pp. 71-85: Springer. Available: https://doi.org/10.1007/978-3-319-22389-6_6.

[15] Thomas Lenz and Lukas Alber, "Towards cross-domain eID by using agile mobile authentication," in *2017 IEEE Trustcom/BigDataSE/ICESS*, 2017, pp. 570-577: IEEE. Available: https://doi.org/10.1109/Trustcom/BigDataSE/ICESS.2017.286.

[16] Kevin Theuermann, Arne Tauber, and Thomas Lenz, "Mobile-only solution for server-based qualified electronic signatures," in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1-7: IEEE. Available: https://doi.org/10.1109/ICC.2019.8762076.

[17] Thomas Zefferer and Peter Teufl, "Leveraging the adoption of mobile eid and e-signature solutions in europe," in *International Conference on Electronic Government and the Information Systems Perspective*, 2015, pp. 86-100: Springer. Available: https://doi.org/10.1007/978-3-319-22389-6_7.

[18] Herbert Leitold and Daniel Konrad, "Qualified Remote Signatures–Solutions, its Certification, and Use," in *Proceedings of 29th SmartCard Workshop*, 2019, pp. 219-231. Available: https://graz.pure.elsevier.com/en/publications/qualified-remote-signatures-solutions-its-certification-and-use.

[19] Ahto Buldas *et al.*, "Server-supported RSA signatures for mobile devices," in *European Symposium on Research in Computer Security*, 2017, pp. 315-333: Springer. Available: https://doi.org/10.1007/978-3-319-66402-6_19.

[20] Elvīra Krēķe, "Electronic identity verification: personal data protection challenges and risks," Master's thesis, Riga Graduate School of Law, 2020. Available: https://dspace.lu.lv/dspace/handle/7/52519.

[21] Philip MacKenzie and Michael K Reiter, "Networked cryptographic devices resilient to capture," *International Journal of Information Security*, vol. 2, no. 1, pp. 1-20, 2003. Available: https://doi.org/10.1007/s10207-003-0022-8.

[22] Kärt Ilja, "Intercepting Network Traffic of the Smart-ID Android Application," Bachelor's thesis, Insitute of Computer Science, Univeristy of Tartu, 2020. Available: https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=69762&year=2020.

[23] Mohammad Hasan Samadani, Mehdi Shajari, and Mohammad Mehdi Ahaniha, "Self-proxy mobile signature: A new client-based mobile signature model," in *2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops*, 2010, pp. 437-442: IEEE. Available: https://doi.org/10.1109/WAINA.2010.125.

[24] Antonio Ruiz-Martínez, Juan Sánchez-Montesinos, and Daniel Sánchez-Martínez, "A mobile network operator-independent mobile signature service," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 294-311, 2011. Available: https://doi.org/10.1016/j.jnca.2010.07.003.

[25] Jakub Breier and Adam Pomothy, "Qualified Electronic Signature via SIM Card Using JavaCard 3 Connected Edition Platform," in *2014 Ninth International Conference on Availability, Reliability and Security*, 2014, pp. 349-355: IEEE. Available: https://doi.org/10.1109/ARES.2014.53.

[26] Glaidson Menegazzo Verzeletti, Emerson Ribeiro de Mello, and Michelle Silva Wangham, "A National Mobile Identity Management Strategy for Electronic Government Services," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 2018, pp. 668-673: IEEE. Available: https://doi.org/10.1109/TrustCom/BigDataSE.2018.00098.

[27] Ingrid Pappel *et al.*, "Systematic digital signing in estonian e-government processes," in *Transactions on large-scale data-and knowledge-centered systems XXXVI*: Springer, 2017, pp. 31-51. Available: https://doi.org/10.1007/978-3-662-56266-6_2.

[28] Evgenia Pisko, "Mobile electronic signatures: progression from mobile service to mobile application unit," in *International Conference on the Management of Mobile Business (ICMB 2007)*, 2007, pp. 6-6: IEEE. Available: https://doi.org/10.1109/ICMB.2007.45.

[29] Florian Otterbein, Tim Ohlendorf, and Marian Margraf, "The german eID as an authentication token on android devices," *arXiv preprint arXiv:1701.04013*, 2017. Available: https://doi.org/10.48550/arXiv.1701.04013.

[30] Somchart Fugkeaw and Pattavee Sanchol, "Proxy-Assisted Digital Signing Scheme for Mobile Cloud Computing," in *2021 13th International Conference on Knowledge and Smart Technology (KST)*, 2021, pp. 78-83: IEEE. Available: https://doi.org/10.1109/KST51265.2021.9415816.

[31] Massimo Pedroli *et al. Overview of Member States' eID strategies*. (2021). Available: https://ec.europa.eu/cefdigital/wiki/display/EIDCOMMUNITY/National+Strategies

[32] eParaksts. *eParaksts mobile*. (06.04.2022). Available: https://www.eparaksts.lv/en/Produkti/Privatpersonam/eParakstsLV/mapp_apraksts

[33] eParaksts. *How to apply for eParaksts mobile*. (11.05.2022). Available: https://www.eparaksts.lv/en/help/faq/Start_using_eParaksts_mobile/How_to_obtain_eParaksts_mobile

[34] VAS LVRTC, "General Terms and Conditions of the Trust Services v.09.0," 19.07.2021. Available: https://www.eparaksts.lv/download/2021_2_2_1_Noteikumi_LVRTC_Uzticamibas_pakalpojumu_sniedzeja_visparejie_09_EN_05072021_162945.pdf

[35] Paul Baka, "Protecting against Man-In-The-Middle Attacks," 13.05.2019. Available: https://www.thesslstore.com/blog/protecting-against-man-in-the-middle-attacks/

[36] Cloudflare. *What is Transport Layer Security (TLS)?* (11.05.2022). Available: https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/

[37] Android Developers. *Security with HTTPS and SSL*. (11.05.2022). Available: https://developer.android.com/training/articles/security-ssl

[38] Jeffery Walton *et al. Certificate and Public Key Pinning*. (11.05.2022). Available: https://owasp.org/www-community/controls/Certificate_and_Public_Key_Pinning

[39] Kinza Yasar, "man-in-the-middle attack (MitM)," 01.04.2022. Available: https://www.techtarget.com/iotagenda/definition/man-in-the-middle-attack-MitM

[40] Mitmproxy. *Modes of Operation*. (11.05.2022). Available: https://docs.mitmproxy.org/stable/concepts-modes/

[41] Tudor Dan, "How to set an Android proxy server for Wi-Fi networks," 07.02.2022. Available: https://www.digitalcitizen.life/how-add-proxy-server-wireless-connection-android/

[42] Mitmproxy. *Transparent proxying*. (11.05.2022). Available: https://docs.mitmproxy.org/stable/howto-transparent/

[43] Mitmproxy. *About Certificates*. (11.05.2022). Available: https://docs.mitmproxy.org/stable/concepts-certificates/

[44] APKLab. *APKLab Documentation*. (11.05.2022). Available: https://github.com/APKLab/APKLab

[45] Volodymyr Hudyma, "Open Mobile Application From The Browser," 30.01.2021. Available: https://vhudyma-blog.eu/open-mobile-application-from-the-browser/

[46] Mohamed Sabt, Mohammed Achemlal, and Abdelmadjid Bouabdallah, "Trusted execution environment: what it is, and what it is not," in *2015 IEEE Trustcom/BigDataSE/ISPA*, 2015, vol. 1, pp. 57-64: IEEE. Available: https://doi.org/10.1109/Trustcom.2015.357.

[47] Apple. *Secure Enclave*. (11.05.2021). Available: https://support.apple.com/guide/security/secure-enclave-sec59b0b31ff/web

[48] VAS LVRTC, "“eParaksts” Trust service policy v.2.2," 11.01.2021. Available: https://www.eparaksts.lv/download/2020_2_1_2_Politika_LVRTC_Uzticamibas_pakalpojuma_eParaksts_sniegsanas_politika_EN_2_2pdf_2021-02-11_144643

[49] XDA Developers, "Google’s Remote Key Provisioning will be mandated in Android 13: What that means for you," 04.04.2022. Available: https://www.xda-developers.com/google-remote-key-provisioning/

[50] Entrust Datacard. *TrustedX eIDAS Platform*. (11.05.2022). Available: https://www.entrust.com/-/media/documentation/datasheets/trustedx-eidas-platform-ds.pdf

[51] Ken Kadet, "Entrust Datacard Acquires Barcelona-Based Safelayer," 08.11.2018. Available: https://www.entrust.com/newsroom/press-releases/2018/entrust-datacard-acquires-barcelona-based-safelayer#:~:text=8%2C%202018)%E2%80%94Entrust%20Datacard,%2Dfactor%20authentication%2C%20electronic%20signature%2C

[52] S.A. Safelayer Secure Communications. *TrustedX eIDAS 4..1.8.0 Integration documentation*. (01.04.2022). Available: https://support.safelayer.com/docs/TX/v4.1.8/TX_EIDAS_INTEGRATION/en/html/115910098.html

[53] S.A. Safelayer Secure Communications. *KeyOne 4.0 Security Target* (2014). Available: https://www.commoncriteriaportal.org/files/epfiles/KeyOne%204.0%20Security%20Target.pdf

[54] Entrust Datacard. *Mobile ID*. (11.05.2022). Available: https://www.entrust.com/-/media/documentation/datasheets/trustedx-eidas-mobile-id-ds.pdf

[55] Android Developers. *Android keystore system*. (11.05.2022). Available: https://developer.android.com/training/articles/keystore

[56] David Cooper *et al.*, "Internet X. 509 public key infrastructure certificate and certificate revocation list (CRL) profile," 2070-1721, 2008, Available: https://www.rfc-editor.org/rfc/rfc5280.html, Accessed on: 11.05.2022

[57] eParaksts, "VAS LVRTC - Integration Guidelines," 30.10.2020. Available: https://wiki.eparaksts.lv/pages/viewpage.action?pageId=5603364

[58] VRAA. *e-address home page (in Latvian)*. (11.05.2022). Available: https://mana.latvija.lv/e-adrese/

[59] *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*, 2014. Available: http://data.europa.eu/eli/reg/2014/910/oj.

[60] Republic of Latvia, "Notification Form For Electronic Identity Scheme Under Article 9 (5) Of Regulation (EU) No. 910/2014," 15.11.2018. Available: https://ec.europa.eu/digital-building-blocks/wikis/display/EIDCOMMUNITY/Latvia?preview=/77370111/148898035/LV_notification%20form%20for%20eID%20scheme.pdf

[61] The Cooperation Network, "Opinion No.7 of the Cooperation Network on the Latvian eID scheme," 02.10.2019. Available: https://ec.europa.eu/digital-building-blocks/wikis/pages/viewpage.action?pageId=148898039

[62] S.A. Safelayer Secure Communications. *Security Target - TrustedX*. (2010). Available: https://www.commoncriteriaportal.org/files/epfiles/2009-03-DS.pdf

[63] Common Criteria portal. *Archived Certified Products*. (11.05.2022). Available: https://www.commoncriteriaportal.org/products/index.cfm?archived=1

[64] European Commission. *Qualified Signature/Seal Creation Devices and Secure Signature Creation Devices*. (11.05.2022). Available: https://esignature.ec.europa.eu/efda/notification-tool/#/screen/browse/list/QSCD_SSCD

[65] The Supervisory Committee of Digital Security. *List of qualified trust service providers and their qualified trust services (in Latvian)*. (11.05.2022). Available: https://www.mod.gov.lv/sites/mod/files/document/20210212_UPS%20saraksts.pdf

[66] QSCert, "Certificate No. P-10171/21-1 regarding compliance of the service eSignature, eSignature eID, eSignature card, eSignature card+," 14.06.2021. Available: https://www.eparaksts.lv/download/Sertifikats_eIDAS_eID_eParaksts_karte_(1)pdf_2021-06-29_231804

[67] Tim Cooijmans, Joeri de Ruiter, and Erik Poll, "Analysis of Secure Key Storage Solutions on Android," presented at the Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices, Scottsdale, Arizona, USA, 2014. Available: https://doi.org/10.1145/2666620.2666627

[68] Di Shen, "Exploiting trustzone on android," *Black Hat USA*, vol. 2, pp. 267-280, 2015. Available: https://www.blackhat.com/docs/us-15/materials/us-15-Shen-Attacking-Your-Trusted-Core-Exploiting-Trustzone-On-Android-wp.pdf.

[69] David Cerdeira *et al.*, "Sok: Understanding the prevailing security vulnerabilities in trustzone-assisted tee systems," in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 1416-1432: IEEE. Available: https://doi.org/10.1109/SP40000.2020.00061.

[70] Fabian Fleischer, Marcel Busch, and Phillip Kuhrt, "Memory corruption attacks within Android TEEs: a case study based on OP-TEE," presented at the Proceedings of the 15th International Conference on Availability, Reliability and Security, Virtual Event, Ireland, 2020. Available: https://doi.org/10.1145/3407023.3407072

[71] eParaksts. *What to do, when e-Identity approval notice suddenly is displayed on the smart device*. (11.05.2022). Available: https://www.eparaksts.lv/en/help/faq/Security/What_to_do,_when_e-Identity_approval_notice_suddenly_is_displayed_on_the_smart_device

[72] eParaksts. *What to do if contact information has changed*. (11.05.2022). Available: https://www.eparaksts.lv/en/help/faq/Using_eParaksts_mobile/What_to_do_if_contact_information_has_changed

[73] A-SIT, "A-SIT-VIG-18-048 Qualified Signature and Seal Creation Device (QSCD) Safelayer's qualified remote electronic signature and seal creation device ("TrustedX eIDAS"), version 4.1.6.0," 29.10.2018. Available: https://www.a-sit.at/downloads/1071

# Appendix

## I. Modifications of the *eParaksts mobile* app by *APKLab*

Below are screen snippets of the *eParaksts mobile* app's source code that was modified by *APKLab* to disable the certificate pinning.

smali_classes2 > okhttp3 > CertificatePinner.smali > check(String, java.util.List)

```
.method public final check(Ljava/lang/String;Ljava/util/List;)V
    # inserted by apk-mitm to disable certificate pinning
    .locals 0
    return-void

    # commented out by apk-mitm to disable old method body
    # .locals 1
    # .annotation system Ldalvik/annotation/Signature;
    #     value = {
    #         "(",
    #         "Ljava/lang/String;",
    #         "Ljava/util/List<",
    #         "+",
    #         "Ljava/security/cert/Certificate;",
    #         ">;)V"
    #     }
    # .end annotation
    #
    # .annotation system Ldalvik/annotation/Throws;
    #     value = {
    #         Ljavax/net/ssl/SSLPeerUnverifiedException;
    #     }
    # .end annotation
    #
    # const-string v0, "hostname"
    #
    # invoke-static {p1, v0}, Lkotlin/jvm/internal/Intrinsics;->checkNotNullParameter(Ljava/lang/Object;Ljava/lang/String;)V
    #
    # const-string v0, "peerCertificates"
    #
    # invoke-static {p2, v0}, Lkotlin/jvm/internal/Intrinsics;->checkNotNullParameter(Ljava/lang/Object;Ljava/lang/String;)V
    #
    # .line 150
    # new-instance v0, Lokhttp3/CertificatePinner$check$1;
    #
    # invoke-direct {v0, p0, p2, p1}, Lokhttp3/CertificatePinner$check$1;-><init>(Lokhttp3/CertificatePinner;Ljava/util/List;Ljava/lang/String;)V
    #
    # check-cast v0, Lkotlin/jvm/functions/Function0;
    #
    # invoke-virtual {p0, p1, v0}, Lokhttp3/CertificatePinner;->check$okhttp(Ljava/lang/String;Lkotlin/jvm/functions/Function0;)V
    #
    # return-void
.end method
```

*Figure 46. Modifications in method check(String, java.util.list) located in ~/smali_classes/okhttp3/CertificatePinner.smali*

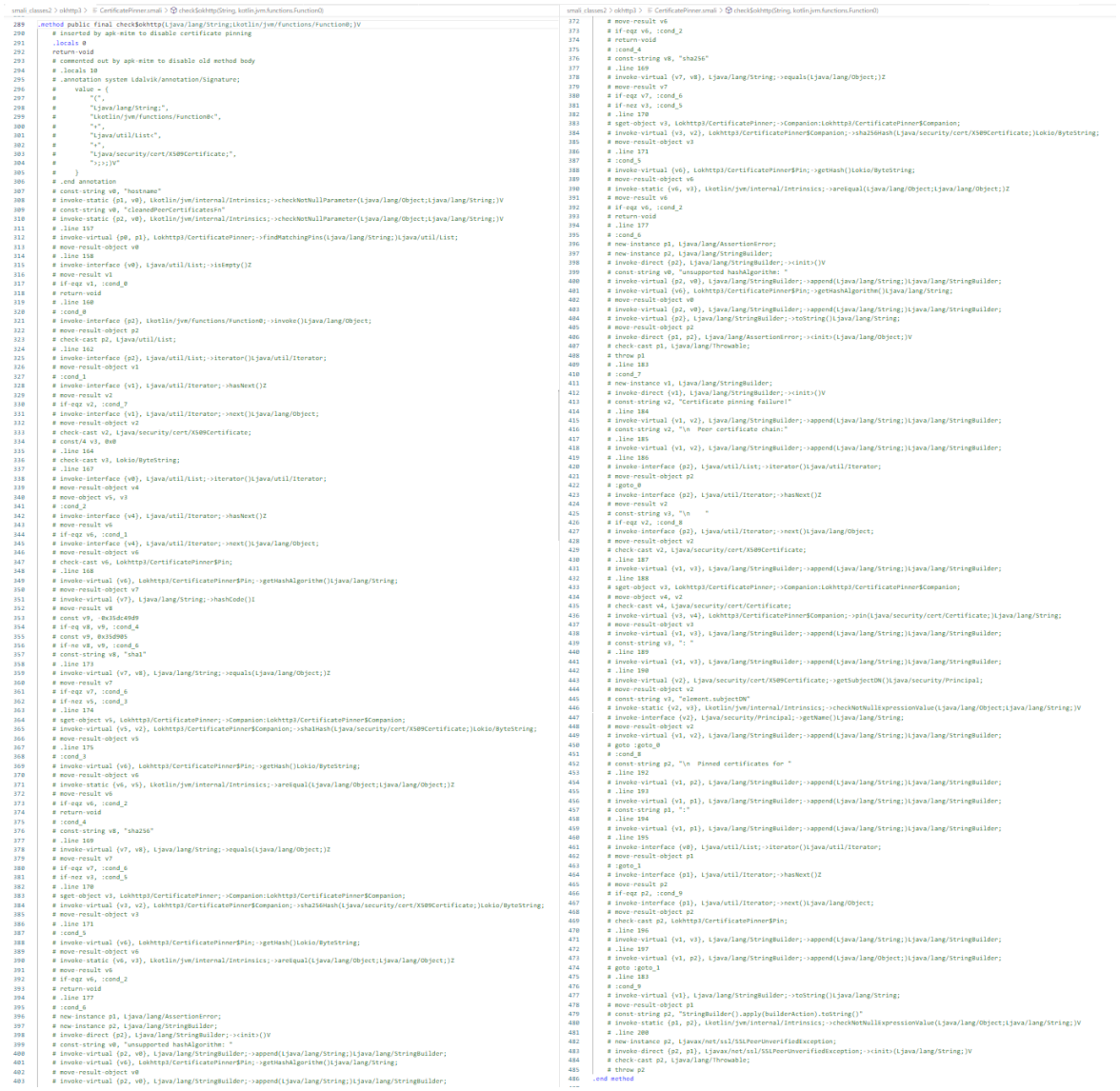

*Figure 47. Modifications in method check$okhttp(String; Function located in ~/smali_classes/okhttp3/CertificatePinner.smali located in)*

smali_classes2 > okhttp3 > internal > tls > OkHostnameVerifier.smali > verify(String, javax.net.ssl.SSLSession)

```
.method public verify(Ljava/lang/String;Ljavax/net/ssl/SSLSession;)Z
    # inserted by apk-mitm to disable certificate pinning
    .locals 1
    const/4 v0, 0x1
    return v0
    # commented out by apk-mitm to disable old method body
    #
    # .locals 1
    #
    # const-string v0, "host"
    #
    # invoke-static {p1, v0}, Lkotlin/jvm/internal/Intrinsics;->checkNotNullParameter(Ljava/lang/Object;Ljava/lang/String;)V
    #
    # const-string v0, "session"
    #
    # invoke-static {p2, v0}, Lkotlin/jvm/internal/Intrinsics;->checkNotNullParameter(Ljava/lang/Object;Ljava/lang/String;)V
    #
    # const/4 v0, 0x0
    #
    # .line 40
    # :try_start_0
    # invoke-interface {p2}, Ljavax/net/ssl/SSLSession;->getPeerCertificates()[Ljava/security/cert/Certificate;
    #
    # move-result-object p2
    #
    # aget-object p2, p2, v0
    #
    # if-eqz p2, :cond_0
    #
    # check-cast p2, Ljava/security/cert/X509Certificate;
    #
    # invoke-virtual {p0, p1, p2}, Lokhttp3/internal/tls/OkHostnameVerifier;->verify(Ljava/lang/String;Ljava/security/cert/X509Certificate;)Z
    #
    # move-result v0
    #
    # goto :goto_0
    #
    # :cond_0
    # new-instance p1, Ljava/lang/NullPointerException;
    #
    # const-string p2, "null cannot be cast to non-null type java.security.cert.X509Certificate"
    #
    # invoke-direct {p1, p2}, Ljava/lang/NullPointerException;-><init>(Ljava/lang/String;)V
    #
    # throw p1
    # :try_end_0
    # .catch Ljavax/net/ssl/SSLException; {:try_start_0 .. :try_end_0} :catch_0
    #
    # :catch_0
    # :goto_0
    # return v0
.end method
```

*Figure 48. Modifications in method verify(String, SSLSession) located in ~/ smali_classes/okhttp3/internal/tls/OkHostnameVerifier.smali*

res > xml > nsc_mitm.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Intentionally lax Network Security Configuration (generated by apk-mitm) -->
<network-security-config>
    <!-- Allow cleartext traffic -->
    <base-config cleartextTrafficPermitted="true">
        <trust-anchors>
            <!-- Allow user-added (proxy) certificates -->
            <certificates src="user" overridePins="true"/>
            <certificates src="system" overridePins="true"/>
        </trust-anchors>
    </base-config>
</network-security-config>
```

*Figure 49. Network configuration file*

## II. Code snippets from the source code of the *eParaksts mobile* app

Below are code snippets of the source code of the *eParaksts mobile* app that was decompiled to Java using *APKLab*.

```
/* renamed from: a */
    private PublicKey m1126a(C1648a.C1650b bVar) throws Exception {
        KeyPairGenerator instance = KeyPairGenerator.getInstance("RSA", "AndroidKeyStore");
        KeyGenParameterSpec.Builder builder = new KeyGenParameterSpec.Builder(
        "com.safelayer.cryptoStore.CipherDataManager_TAG_cipher_key_alias", 3);
        builder.setKeySize(2048).setEncryptionPaddings(new String[]{"PKCS1Padding"});
        instance.initialize(builder.build());
        KeyPair generateKeyPair = instance.generateKeyPair();
        PrivateKey privateKey = generateKeyPair.getPrivate();
        CipherData cipherData = new CipherData();
        cipherData.mo21142a(new KeyInfo(((android.security.keystore.KeyInfo) KeyFactory.getInstance(privateKey.
        getAlgorithm(), "AndroidKeyStore").getKeySpec(privateKey, android.security.keystore.KeyInfo.class)).
        isInsideSecureHardware() ? "and_hardware" : "and_software", "RSA", 2048));
        cipherData.mo21145b("android_keystore");
        cipherData.mo21143a("best-effort");
        bVar.mo21086a(CipherData.f877a, cipherData.mo21141a());
        return generateKeyPair.getPublic();
    }
```

*Figure 50. Code snippet of a method used for key pair generation in Android keystore located in ~sources/com/safelayer/identity/impl/store/upgrade/versions/DataVersion06.java*

```
    private VerificationPinViewState verifyPin(int[] iArr) {
        try {
            return this.pinManager.verify(ArrayUtils.toBytes(iArr)) ? VerificationPinViewState.Finished.Validated : invalidPinStates();
        } catch (Exception e) {
            return new VerificationPinViewState.Error(e);
        }
    }
```

*Figure 51. Code snippet of method "VerifyPin()" located in ~/sources/com/safelayer/mobileidlib/verificationpin/verificationPinViewModel.java*

## III. Example of XAdES signature

The below XML structure is an example of XAdES signature that was extracted from Base64 encoded value embedded in request in step 12 in Figure 33.

```
ss<?xml version="1.0" encoding="UTF-8"?>
<Transaction>
    <Content Id="Service" MimeType="text/xml">
        <Service Domain="urn:safelayer:eidas:domain:oauth:client"
Id="eparaksts_portal" Name="eParaksts"/>
    </Content>
    <Content Id="SignatureProcess" MimeType="text/xml">
        <SignatureProcess Domain="citizen" Id="alcb6e8jcmcd36hs2uo7npje6t"
OperationType="authentication"/>
    </Content>
    <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#" Id="Signature">
        <ds:SignedInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
            <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-
exc-c14n#"/>
            <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-
more#rsa-sha256"/>
            <ds:Reference Id="Reference-Service" URI="#Service">
                <ds:Transforms>
                    <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"/>
                </ds:Transforms>
                <ds:DigestMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>

<ds:DigestValue>/7776yNDoKuZPtytWUatAsha6fJZ1B8E7EK8PnJSIHs=</ds:DigestValue>
            </ds:Reference>
            <ds:Reference Id="Reference-SignatureProcess"
URI="#SignatureProcess">
                <ds:Transforms>
                    <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"/>
                </ds:Transforms>
                <ds:DigestMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>

<ds:DigestValue>RERSn9+b1tgP1EvGxRXbN1IuJOyvlBIQeKI4ab/OW8I=</ds:DigestValue>
            </ds:Reference>
            <ds:Reference Type="http://uri.etsi.org/01903#SignedProperties"
URI="#Xades">
                <ds:Transforms>
                    <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"/>
                </ds:Transforms>
                <ds:DigestMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>

<ds:DigestValue>eDDhxk4s7/zjpPuRN/XRHZEFrkEUWhe5f54IqD/8O+o=</ds:DigestValue>
            </ds:Reference>
        </ds:SignedInfo>

<ds:SignatureValue>b+R4J0BjMCTWvWJ9IEoVFYPsTtECewc6ptKMHYVk4UtQjr5SECoqOcciyXi8E
/2TI0Yui0ntOvs2gQsmikw5i42chyeR8KC2UGbQe8ntXphk/lLCkRwsFf8AIK3rhUhfUKH5nQZ/EghZ2
P7TFkCUzfLYsEfWQyv3KOdZ789JuScQzW4OoBmr7cERAB7/5ncUv0L0eQv+kHn7FQiMSizsy1Ho2DWuS
vEomANc2KLvRY9e+eghMh/fUHp4YuJVYy/whNN6fN8VdV8z2ufEpY7dh8XO2ti19pvSARI9QfrmOk/8O
CiBnMv8rx//SVyqJ9UOjx22OpXBkGkU/L+uuI5pMA==</ds:SignatureValue>
        <ds:KeyInfo>
            <ds:X509Data>
                <ds:X509Certificate>[Authentication
certificate]</ds:X509Certificate>
            </ds:X509Data>
```

```
        </ds:KeyInfo>
        <ds:Object>
            <xades:QualifyingProperties
xmlns:xades="http://uri.etsi.org/01903/v1.3.2#" Target="#Signature">
                <xades:SignedProperties
xmlns:xades="http://uri.etsi.org/01903/v1.3.2#" Id="Xades">
                    <xades:SignedSignatureProperties>
                        <xades:SigningTime>2022-03-
28T14:38:11Z</xades:SigningTime>
                        <xades:SigningCertificate>
                            <xades:Cert>
                                <xades:CertDigest>
                                    <ds:DigestMethod
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
Algorithm="http://www.w3.org/2001/04/xmlenc#sha256"/>
                                    <ds:DigestValue
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">c60uNV9EFxnWZJwe4iZLhvZObUz/0f24mj
PVtLisUXQ=</ds:DigestValue>
                                </xades:CertDigest>
                                <xades:IssuerSerial>
                                    <ds:X509IssuerName
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">CN=LV eID ICA
2021,2.5.4.97=#0c114e54524c562d343030303033303131323033,O=VAS Latvijas Valsts
radio un televīzijas centrs,C=LV</ds:X509IssuerName>
                                    <ds:X509SerialNumber
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">804573203914238030038517307055564
3872</ds:X509SerialNumber>
                                </xades:IssuerSerial>
                            </xades:Cert>
                        </xades:SigningCertificate>
                    </xades:SignedSignatureProperties>
                    <xades:SignedDataObjectProperties>
                        <xades:DataObjectFormat ObjectReference="#Reference-
Service">
                            <xades:MimeType>text/xml</xades:MimeType>
                        </xades:DataObjectFormat>
                        <xades:DataObjectFormat ObjectReference="#Reference-
SignatureProcess">
                            <xades:MimeType>text/xml</xades:MimeType>
                        </xades:DataObjectFormat>
                    </xades:SignedDataObjectProperties>
                </xades:SignedProperties>
            </xades:QualifyingProperties>
        </ds:Object>
    </ds:Signature>
</Transaction>
```

## IV. License

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

I, **Elizabete Liene Šterna**,

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,
   **Security Architecture of the Latvian eParaksts mobile,**
   supervised by Arnis Paršovs.

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.

**Elizabete Liene Šterna**

**17.05.2022**