Security Analysis of RIA's Authentication Service TARA

Arnis Parsovs
University of Tartu

May 28, 2021

1 Introduction

The Estonian Information System Authority (Riigi Infosüsteemi Amet – RIA) provides a federated authentication solution TARA that can be used by public sector institutions to authenticate ID card, Mobile-ID and Smart-ID and eID users from other EU Member States [1].

This report discusses findings from the security analysis of TARA that was performed in May 2021. The analysis was based on the TARA technical specification v1.8 2021-03-19 available at [2]. The analysis focused on the protocol flow between service providers and TARA. The security of the authentication flows on the TARA side (i.e., how authentication using different authentication methods is implemented by TARA) was not in the scope of this analysis.

1.1 TARA authentication protocol

The protocol implemented by the TARA authentication service is based on the OpenID Connect protocol [3]. The protocol involves 3 parties: a service provider who wants to authenticate a user, the user that gets authenticated and the TARA server that performs authentication of the user using various eID tools. The high-level protocol flow is depicted in Figure 1.

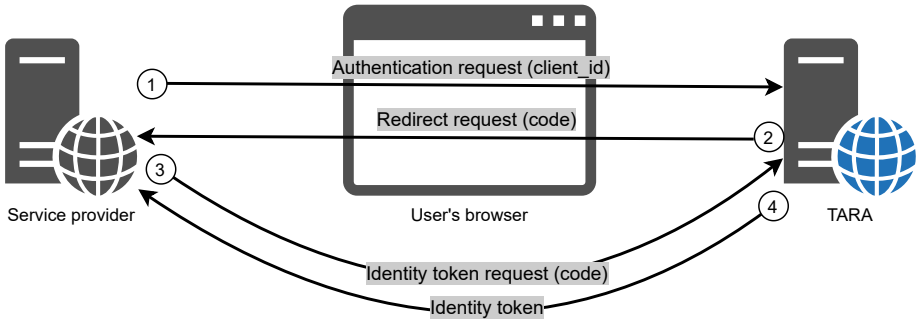


Figure 1: TARA authentication protocol

The authentication process starts with a user initiating a log in process in a service provider's website. The service provider redirects the user's browser to the TARA website together with an authentication request that among other things contains the service provider's identifier (the `client_id` field). After the user has performed a successful authentication at the TARA website, the user's browser is redirected back to the service provider's website together with a redirect request[1] that among other things contains an authorization code (the `code` field). The service provider makes a direct connection to the TARA server and sends an identity token request that among other things contains the authorization code received in the redirect request. As a response, the TARA server returns a cryptographically signed identity token that contains the personal data of the user. The service provider verifies the authenticity of the identity token and based on the data therein authenticates the user.

2 Findings

In the subsections below, we discuss the issues that were found in the course of the analysis. When possible, we provide specific recommendations on how to solve them. The issues listed below have been ordered based on our opinion of their significance.

2.1 Authenticity of the identity token

The identity token issued by TARA is cryptographically signed by the TARA authentication service using the SHA-256 hash function with a 4096-bit RSA key. The public key that service providers must use to verify the signature is served by the same TARA server that returns the signed identity token. Furthermore, the technical specification of TARA explicitly states that service providers should not hardcode the public key (see Section 5.1.1 in [2]).

We find that in this configuration the signing of the token provides no security benefit. This is so because an attacker, who is able to man-in-the-middle the connection between a service provider and the TARA server, will be able to sign tokens using the attacker's own private key and provide the corresponding public key for verification to the unsuspecting service provider.

Hence, in the current configuration the verification of the token's signature is irrelevant as the security of the entire process relies on a service provider correctly authenticating the TARA server when sending the identity token request. To authenticate the TARA server, the TLS library used by the service provider has to verify the public key certificate of the TARA server.

The technical specification of TARA correctly notes that the service provider has to verify the TARA server certificate. More specifically, it states that the certificate must be verified by "using DigiCert's root certificate or TARA certificate as a trust anchor" (Section 5.1.2 in [2]). We point out, however, that there are a list of things that can go wrong with the verification of the TARA server certificate:

[1] The TARA technical specification uses the term "redirect request" to denote the authentication response from TARA. In our opinion, the term "authentication response" used in the OpenID Connect specification [3] is a more intuitive term to describe this request.

1. First of all, there is a large amount of evidence that indicates, that in practice, it is very difficult to implement TLS certificate validation in non-browser software correctly [4]. Usually, correct server certificate validation requires extra effort from a software developer, and therefore it is common to see that no certificate validation is performed, which results in any certificate offered by the server being accepted as valid.

2. By default, it is common for TLS libraries to verify a server certificate using Certificate Authorities (CAs) that are defined in the operating system trust store. The recommendation to use the DigiCert's root CA certificate as a trust anchor is helpful, as this narrows down the number of trusted parties (CAs) that the security of TARA relies upon. We note, however, that using the DigiCert's root CA certificate as a trust anchor does not eliminate the risk that a lower-assurance domain validated[2] certificate of TARA obtained by an attacker would be accepted as valid. The recommendation to hardcode the TARA certificate as a trust anchor would eliminate this risk, but has a drawback that service providers have to update their implementations whenever the TARA server certificate is renewed (which is at least every 398 days[3]).

3. Even if the TARA server certificate is hardcoded as a trust anchor, there is an issue of ensuring adequate secrecy of the corresponding private key. The current certificate[4] used by the server `tara.ria.ee` is a wildcard certificate that is also used on a list of other RIA servers administered by possibly different (hopefully RIA) employees. This means that if any of these servers are compromised to the extent that an attacker obtains the private key corresponding to this certificate, the attacker can execute successful man-in-the-middle attacks effectively bypassing authentication in any service provider that uses TARA.

To summarize: (1) service providers' implementation of the TARA server certificate validation is susceptible to different types of failures; (2) the TARA server certificate validation as recommended by RIA requires hardcoding of the TARA server certificate, which has to be regularly rotated; (3) it is much harder to ensure the secrecy of the TARA server's private key that is shared among different RIA servers, compared to ensuring the secrecy of a single-purpose private key that is used for TARA identity token signing.

Based on the abovementioned, we recommend to RIA to remove the requirement for the DigiCert's root CA and TARA server certificate hardcoding, but instead require service providers to hardcode the public key that is used by TARA to sign the identity tokens.

Implementing this recommendation would involve the removal of the TARA public signature key endpoint, instead distributing the TARA public key to service providers using some out-of-band means, such as sending it to the service providers digitally signed by a representative of RIA.

[2]The current TARA server certificate is organization validated, meaning that it has been issued based on a higher identity validation process than a domain validated certificate would.

[3]There are indications that the maximum lifespan of the TLS server certificate validity in the future will decrease even further.

[4]`https://crt.sh/?id=3288095677`

The benefit of this approach is that it would remove the reliance on Web-PKI from the trust assumptions, would provide RIA flexibility to decide on rotation of the trust anchor (the token signing key in this case) and would provide viable means for RIA to protect the private key of the trust anchor.

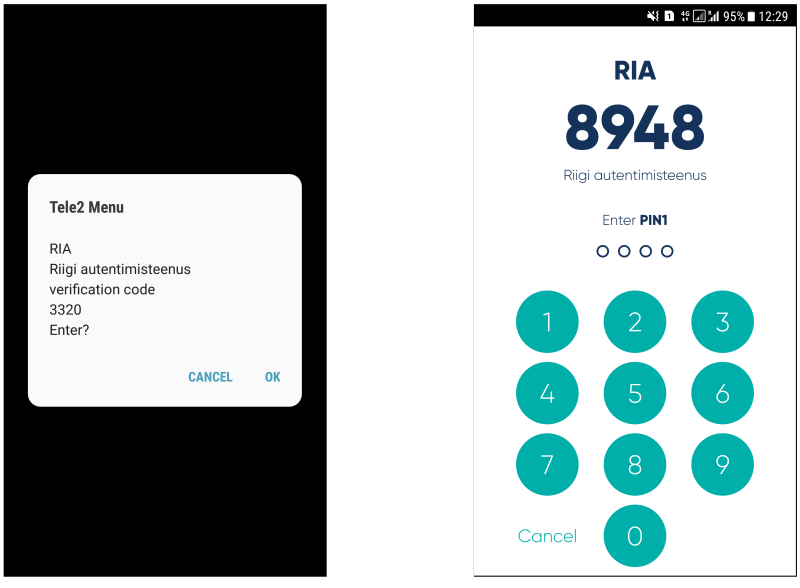
2.2 Susceptibility to Mobile-ID and Smart-ID phishing attacks

We have observed that regardless of the service provider to which a user is authenticating via TARA, the Mobile-ID and Smart-ID authentication prompts shown in the user's device always display the service provider identifier "RIA Riigi autentimisteenus" (see Figure 2).

The use of such a generic service provider identifier makes the Mobile-ID and Smart-ID authentication process susceptible to phishing attacks. An attacker can set up a malicious website that requires authentication using Mobile-ID or Smart-ID. When a victim enters their Mobile-ID or Smart-ID identifier in the malicious website, the attacker forwards it to TARA and shows to the victim the correct Mobile-ID/Smart-ID verification code returned by TARA. If the victim fails to notice that the Mobile-ID or Smart-ID identifier was not entered in the authentic TARA environment and confirms the Mobile-ID/Smart-ID authentication request on their mobile device, the attacker will be able to authenticate on behalf of the victim to any service provider that uses TARA.

The application for joining the TARA service [5] requires service providers to specify a short name, and the TARA documentation provides recommendations on how to choose it [6]. However, for some reason this identifier is not used in practice.

We recommend to RIA to display the service provider's short name on the user's mobile device when the user authenticates using Mobile-ID and Smart-ID.



(a) Mobile-ID (b) Smart-ID

Figure 2: Authentication prompt on a user's device when authenticating via TARA

2.3 Protection against false login attacks

The technical specification of TARA requires a service provider to include the `state` parameter in the authentication request (Section 4.1 in [2]). The value specified in the `state` parameter is returned by TARA back to the service provider in the redirect request.

The `state` parameter can be used to prevent a cross-site request forgery attack, in which an attacker forces a victim's browser to submit a redirect request to a service provider that contains an authorization code from an authentication process in which the attacker has been authenticated at TARA. If the `state` parameter was not used, the service provider would accept the authorization code and would log the victim's browser into the attacker's account at the service provider's website. This attack is described in the TARA documentation [7], but is not referenced in the technical specification.

Section 5.2 "Protection against false request attacks" of the TARA technical specification describes how service providers should implement the verification of the `state` parameter to prevent the cross-site request forgery attack mentioned above. However, we note that the instructions provided in the specification are suboptimal due to the following reasons:

1. It is stated that protection against the attack can be achieved by using the `state` and `nonce` security codes. However, the instructions on how to use the `nonce` parameter to achieve this protection are not provided. A confusingly similar term "nonce word" is used in the description, while it actually is not related to the `nonce` parameter.

2. It is stated that the `HttpOnly` attribute must be applied to the cookie, but it is not stated that the `Secure` attribute should be applied as well (to prevent the cookie being sent over plaintext HTTP requests).

3. The value specified in the `state` parameter is derived by hashing a random value included in a cookie. It is not explained why the same protection cannot be achieved by putting the random value that is stored in the cookie directly into the `state` parameter.

4. The instructions require a service provider to introduce an additional TARA-specific cookie, while the same can be achieved using the service provider's session mechanism that is already in place to maintain a web session with the user's browser.

We recommend to RIA to improve the instructions on how to use the `state` parameter. We propose the following wording for the instructions:

> The value for the `state` parameter has to contain at least 16 random bytes to prevent the client application from reusing the same `state` value repeatedly and prevent attackers from guessing the value.
>
> The `state` parameter included in the authentication request must be bound to the browser session of the user who initiated the authentication process.
>
> Preferably, the `state` value should be stored on the server side. If it is stored on the browser side, it must be cryptographically protected to prevent a malicious user from modifying the `state` value that is

associated with the user's web session. This will prevent an attack where a valid redirect request issued to a victim can be used in an attacker's browser session to authenticate on behalf of the victim[5].

When receiving the redirect request, the client application must verify that the `state` parameter in the redirect request match the value for the browser session of the user from whom the request was received.

2.4 Verification of the identity token's `state` field

The `state` value sent in the authentication request is present in the cryptographically signed identity token (Section 4.3.1 in [2]). However, the TARA technical specification (Section 5.1 in [2]) does not require a service provider to verify that the `state` field of the identity token contains the same value as the one that was associated with the user's browser session.

We see two attacks that such a check would prevent:

1. An authorization code substitution attack where a valid authorization code issued to a victim can be used in an attacker's browser session to authenticate on behalf of the victim.

2. An identity token substitution attack by an attacker who has hijacked the TLS connection between the service provider and the TARA server[6].

Based on the abovementioned, we recommend to RIA to include the requirement to check the `state` value when verifying the identity token.

2.5 The use of the `nonce` paramater

The specification mentions the `nonce` parameter and that it can help to prevent replay attacks, but does not explain how it should be set and verified.

We note that the TARA server prevents identity token replay attacks as the service providers cannot use the same authorization code more than once[7]. Even if the TARA server tried to return the same identity token more than once, this would fail as the `state` value present in the token would not match the unique state value associated with the authentication session[8]. Hence, the secure implementation of the protocol prevents identity token replay attacks without the need for a service provider to remember the identifiers of the processed identity tokens. If, however, the service provider needs to uniquely identify the token, the mandatory `state` parameter that is included in the identity token can be used for this purpose.

We recommend to RIA to remove the `nonce` parameter from the protocol and specification.

[5] To achieve full protection against such an attack, the check described in Section 2.4 has to be implemented as well.

[6] This attack is relevant only if the recommendation specified in Section 2.1 is implemented.

[7] While not explicitly stated in the TARA technical specification, the TARA test service in practice implements the OpenID Connect recommendation (Section 3.1.3.2 in [3]) by verifying that the authorization code has not been previously used.

[8] Assuming that the check described in Section 2.4 is implemented.

2.6 Protection against false authentication initiation attacks

The protection against false login attacks (Section 2.3) does not protect against false authentication initiation attacks, where a malicious website redirects a victim's browser to the TARA website, specifying a `client_id` in the authentication request that belongs to a legitimate service provider that uses TARA. If the service provider implements protection against false login attacks, the attacker will not be able to trick the victim into authenticating to the service provider. I.e., after the victim has authenticated at the TARA website, the redirect request sent by the victim's browser will be rejected by the service provider. However, if the attacker is more powerful, meaning that the attacker is able to observe the authorization code returned to the victim, the attacker will be able to impersonate the victim on the service provider's website.

To prevent such attacks, the TARA website should clearly show the service provider for which the current authentication process has been initiated. In the current user interface of the TARA website this is not made clear, although the user may try to establish this by looking at the URL that is present in the "Return to service provider" link (see Figure 3).

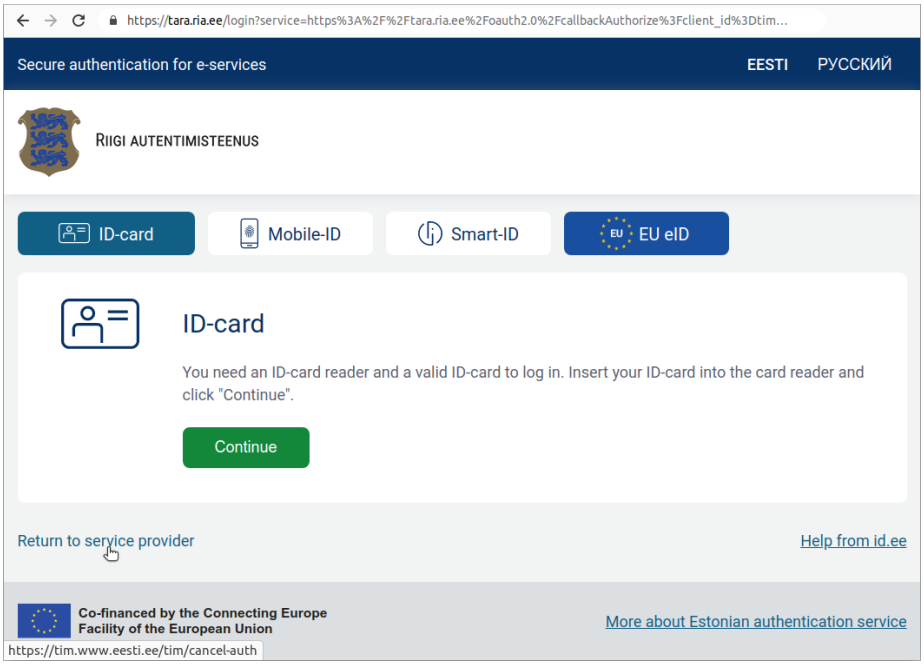


Figure 3: Authentication menu as shown on the TARA website

2.7 Expiration time of the identity token

According to the TARA technical specification (Section 6.3 in [2]), the service provider has to obtain the identity token within 30 seconds after the authorization code has been issued. This requirement is helpful as it guarantees that in case the service provider was able to obtain the identity token, the corresponding authentication process is fresh (i.e., the user has authenticated

at TARA in the last 30 seconds). This, however, questions whether there is a need for further token expiration checks and why the expiration time in the token is set to 10 minutes (Section 6.3 in [2]).

Checking the expiration time of the token is useful as it eliminates the risk of an outdated token being accepted, if for some reason[9] the requirement to obtain the identity token in 30 seconds is not enforced on the TARA side.

The 10 minute expiration time of the token, however, seems excessive as the verification of the token should take place at maximum in a few seconds after the identity token is obtained from TARA using the identity token request. Hence, a 40 second expiration time of the token would seem more meaningful. If the 10 minute expiration time of the token is there to allow service providers to have their system time up to 10 minutes in the future, then this assumption should be explicitly noted.

We recommend to RIA to clarify the reasoning for setting the expiration time of identity tokens to 10 minutes.

2.8 Verification of the validity period of the identity token

The identity token contains two semantically very similar fields: `iat` (Issued At) and `nbf` (Not Before). The `iat` field is described to contain "the time of issue of the certificate"[10], while the `nbf` field is described to contain "the validity start time of the certificate" (Section 4.3.1 in [2]). It is not clear to us what the purpose of the `nbf` field is and why it contains a different value than the `iat` field. Section 5.1.5 "Verifying the validity of the certificate" of the TARA technical specification states that the `iat` and `nbf` fields have to be verified, but then instructs how to verify the `nbf` field without explaining how the `iat` field should be verified.

We recommend to RIA to sort out the difference between these fields and clarify the respective security checks.

2.9 Inclusion of the redirect URL in the requests

The application for joining the TARA service [5] requires service providers to specify a redirect URL, where, after a successful authentication at the TARA side, the user's browser will be redirected together with the authorization code. The redirect URL of the service provider is hardcoded on the TARA side and is looked-up by TARA based on the service provider's `client_id` specified in the authentication request.

According to the current specification, the service provider has to include the same redirect URL in the authentication and identity token requests. This, however, unnecessarily complicates the protocol and introduces the risk that due to a coding error on the TARA side, the redirect URL submitted in a potentially malicious request is used to perform security-critical decisions.

We recommend to RIA to remove the requirement to include the `redirect_uri` field in the protocol requests. Similarly, we recommend reconsidering the necessity to include other parameters that have fixed values (e.g., `response_type` and `grant_type`).

[9] For example, in case of a coding error or misconfiguration on the TARA side, or in the case of a successful man-in-the-middle attack between a service provider and the TARA server.

[10] In our opinion, the term "certificate" used by the TARA technical specification to denote the identity token is confusing.

3 Concluding remarks

We commend the designers of the TARA service for using the authorization code flow to implement the protocol, since the implicit flow (e.g., as used in the bank link protocol [8]) opens the authentication process to a variety of additional security risks. We also commend the authors of the TARA technical specification for explicitly listing the security checks that have to be performed by service providers.

The findings described in this report show that there is room for clarifying the technical specification and for hardening the protocol. By implementing the recommendations listed in this report, the TARA protocol will achieve protection against impersonation attacks even in the presence of an attacker who is: (1) able to observe authentication requests sent by the user's browser to the TARA server; (2) able to observe and modify redirect requests returned by the TARA server to the user's browser; and (3) able to observe and modify the messages exchanged between the service provider and the TARA server.

Acknowledgements. This research has been carried out with financial support from the European Social Fund through the IT Academy programme and from the Estonian Ministry of Economic Affairs and Communications.

References

[1] Estonian Information System Authority. The Information System Authority's authentication services, May 19, 2021. `https://www.ria.ee/en/state-information-system/eid/partners.html`.

[2] Estonian Information System Authority. TARA Technical specification v1.8, March 19, 2021. `https://e-gov.github.io/TARA-Doku/TechnicalSpecification`.

[3] The OpenID Foundation. OpenID Connect Core 1.0 incorporating errata set 1, November 8, 2014. `https://openid.net/specs/openid-connect-core-1_0.html`.

[4] Martin Georgiev, Subodh Iyengar, Suman Jana, Rishita Anubhai, Dan Boneh, and Vitaly Shmatikov. The most dangerous code in the world: validating SSL certificates in non-browser software. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, 2012.

[5] Estonian Information System Authority. Application for joining the test environment of the National Authentication Service (TARA) (in Estonian), August 17, 2020. `https://www.ria.ee/sites/default/files/test_tara_liitumistaotlus_2020_0.pdf`.

[6] Estonian Information System Authority. TARA Documentation: Self-help (in Estonian), May 20, 2021. `https://e-gov.github.io/TARA-Doku/Eneseabi`.

[7] Estonian Information System Authority. TARA Documentation: Counterfeit attack and protection against it (in Estonian), May 20, 2021. `https://e-gov.github.io/TARA-Doku/Volts`.

[8] Arnis Parsovs. *Security Analysis of Internet Bank Authentication Protocols and their Implementations*. MSc thesis, Tallinn University of Technology, 2012. `https://kodu.ut.ee/~arnis/bankauth/thesis.pdf`.

Disclosure

On 2021-05-28, this report was shared with RIA. On 2021-06-04, RIA provided feedback on each finding covered in this report:

- Section 2.1: Authenticity of the identity token - requires a more detailed analysis from RIA whether and how to implement this change.
- Section 2.2: Susceptibility to Mobile-ID and Smart-ID phishing attacks - this has been already implemented in TARA, but several service providers have not provided the short name of their application. As this affects especially those clients, who have made their integration with TARA before RIA made this change, additional communication will be needed to fully resolve this security threat.
- Section 2.3: Protection against false login attacks - RIA needs to first check the compatibility with OpenID Connect profile before implementing this change.
- Section 2.4: Verification of the identity token's state field - RIA will improve the specifications.
- Section 2.5: The use of the nonce parameter - We agree that nonce basically duplicates the state parameter. RIA will remove the nonce parameter from specification. However, we cannot remove nonce from the protocol as it break the interface between TARA and 300+ client applications.
- Section 2.6: Protection against false authentication initiation attacks - this has been already implemented in TARA, but several service providers have not provided the short name of their application. As this affects especially those clients, who have made their integration with TARA before RIA made this change, additional communication will be needed to fully resolve this security threat.
- Section 2.7: Expiration time of the identity token - the reason behind the differences between the values was due to Apereo CAS limitations. In TARA version 2.0 this value can be configured and RIA will implement this change in the upcoming sprints.
- Section 2.8: Verification of the validity period of the identity token - RIA will clarify this in the specifications.
- Section 2.9: Inclusion of the redirect URL in the requests - requires a more detailed analysis from RIA whether and how to implement this change. It needs to be clarified but it is possible that removing the requirement to include the redirect uri field in the protocol requests might result non-compliance with OpenID Connect profile, which we highly rely on. RIA sees that the impact of this risk is also low.