Options for integrating eID and SAML

Detlef Hühnlein
ecsec GmbH
detlef.huehnlein@ecsec.de

Jörg Schwenk
Ruhr Univerität Bochum
joerg.schwenk@rub.de

Tobias Wich
ecsec GmbH
tobias.wich@ecsec.de

Florian Feldmann
Ruhr Univerität Bochum
florian.feldmann@rub.de

Vladislav Mladenov
Ruhr Univerität Bochum
vladislav.mladenov@rub.de

Andreas Mayer
Würth GmbH
andreas.mayer@wuerth.com

Moritz Horsch
Technische Universität Darmstadt
moritz.horsch@cdc.informatik.tu-darmstadt.de

Bud P. Brügger
Fraunhofer IAO
bud.bruegger@iao.fraunhofer.de

Johannes Schmölz
ecsec GmbH
johannes.schmoelz@ecsec.de

ABSTRACT

Several European countries currently introduce highly sophisticated eID functionality in their national identity cards. This functionality typically has no direct relation to web security standards, but will be integrated with web technologies to enable browser-based access to resources.

The research challenge to combine eID protocols and web standards like TLS in a secure way proves extremely challenging: The security of many of the proposed systems boils down to HTTP session cookies and TLS server certificates. The overall security is not improved, which doesn't justify the additional costs.

In this paper, we investigate this security challenge for the German national identity card and its eID functionality. We show that the solution currently standardized by the German government does not offer any additional security, by giving an in-depth analysis of the complete software system. We discuss several possible paths to an enhanced solution, all based on TLS channel bindings. Finally we describe a system setup based on the SAML Holder-of-Key web browser profile, which also mitigates interoperability problems.

1. INTRODUCTION

Governmental IDM. In the area of governmental Identity Management, there seem to be two major trends, which are addressed in publicly funded projects such as STORK 2.0[1], FutureID[2] and SkIDentity[3] for example. First, Federated Identity Management solutions are increasingly used in practice as they allow to implement Single Sign-On (SSO) and facilitate the integration of Service Providers. The Security Assertion Markup Language (SAML), which has been developed by OASIS, plays a central role in the implementation of Federated Identity Management. Second, several European countries have introduced electronic identity cards (eID) on a national level (see [19]) and there seems to be a trend to combine the eID-based authentication and identification with SAML-based identity federation (see [25, 17]).

SSO. Single Sign-On (SSO) systems enable strong authentication for web users: A web application (the Service Provider, SP; sometimes also called Relying Party, RP) may authenticate a user with the help of a specialized authentication server (the Identity Provider, IdP). The IdP may implement a variety of different authentication mechanisms, ranging from passwords to novel cryptographic protocols. Additionally, strong smartcard-based authentication at the IdP can be used to strengthen the authentication process. Subsequently, an authentication token containing the confirmed identity information will be generated by the IdP and transferred to the SP. After the successful verification of the token the SP starts an authenticated session, e.g. via HTTP session cookies, and grants access to the resources.

In this paper we investigate the research issue on how to securely combine different eID technologies with Web standards within a SSO authentication process, in detail for the German eID system [25]. In this system the IdP, providing the authentication, consists of three different server instances (eID server, TCToken Service(TS), and SAML-IdP, see Figure 1). These components are responsible for the eID-authentication and the generation of the SAML-based authentication token. On the client side, the eID authentication is split between a special eID application, activated by the browser, and a smartcard.

[1]See www.eid-stork2.eu/

[2]See www.futureid.eu.

[3]See www.skidentity.de.

Unfortunately, the system involves three non-standard TLS channels, and one TLS channel between browser and SP. Furthermore, if the eID client uses its own TLS connection to the IdP (as it is the case in the German eID system), this connection gets lost, as there is yet no efficient way to securely link this TLS connection with any connection established by the browser. We therefore investigate different possibilities on how a separate eID application can nevertheless use a TLS channel established by the browser.

Main research question. One major research challenge in SSO systems is to reduce the complexity of the authentication process and secure all parts of the communication: the authentication at the IdP, the secure transport of the generated token to the SP and the security of the session after the successful authentication.

As part of our security analysis we assume an attacker controlling all network traffic, which is the standard model in cryptography. This is a very strong model, and it is motivated by the fact that eID cryptographic protocols have been shown to be secure in this model. It allows us to show some generic attacks on the specified systems, and to propose an improved solution. Once the *specified* systems will be *implemented*, it may be possible to derive attacks in a weaker model, e.g. in a web attacker model, where the adversary is only able to lure the victim to a website controlled by him. However, it is impossible to derive any XSS, CSRF or Clickjacking attacks against system specifications, since these attacks target implementation weaknesses.

In order to achieve our goals and approve the security against the chosen adversarial model, we analyzed several TLS channel bindings: Recognition of a web browser through TLS client certificates [39], recognition of the server through the public key contained in the sever certificate (RFC 5929, server-endpoint binding), and recognition of the TLS session through the first FINISHED message (RFC 5929, session-binding). All three bindings can be used to secure SSO systems. However, only client certificates are already fully supported by all major browsers. Thus, no further implementations or installation of additional software on the browser is needed. The major research challenge here is the fact that the eID application is a separate application, and not a browser plugin. SAML-HoK relies on the fact that the IdP records the TLS client certificate used by the web browser, and inserts it after successful authentication into the SAML assertion issued. The SP may then check that the same client certificate was used in its own TLS connection with the browser, and may thus conclude that authentication at the IdP was performed using the same browser[4].

The main question we are going to answer is the following: How can we combine on the client system TLS channel binding with a proprietary authentication mechanism that is not supported by the browser? As our answer, we show a construction where the proprietary algorithm is implemented as a separate application (which also includes the smartcard drivers), but this application uses the browser as a proxy for all network communications.

Contribution. The contribution of this paper is as follows:

- We give an in-depth description of one of the major eID initiatives in Europe and consider its security.
- We propose, based on the generic architecture of the German eID system, a more simple and secure variant for this SSO system, based on the SAML Holder-of-Key web browser profile.

Structure. The rest of the paper is organized as follows: Section 2 provides an overview over the current SAML specifications and related work from the security research. In contrast to that work, eID based standards for authentication and identification are presented as well. An in-depth description of the SAML profile specific to the german nPA is given in Section 3, followed by a technical analysis focusing on security. Section 4 introduces the well researched SAML Holder-of-Key (HoK) binding for the Web Browser SSO profile. Based on this binding a new eID based SAML profile is proposed, which features an optimized message flow and advantages with respect to security and privacy. Section 5 concludes the paper and provides an outlook on possible future improvements.

2. BACKGROUND ON SAML AND EID-ACTIVATION

2.1 Overview of SAML Version 2.0

SAML is a set of standards, which has been developed by the OASIS Security Services Technical Committee[5] and defines the syntax and semantics for XML-encoded assertions about authentication, attributes and authorization together with related protocols that convey such assertions and the binding of these protocols to various transfer protocols. The different versions of SAML (v1.0 [32], v1.1 [42] and v2.0 [9]) have been influenced by previous work at IETF (RFC 2903 [14]) and projects like the Liberty Alliance[6], which is now called Kantara Initiative[7], and Shibboleth[8] and is now adopted by an increasing number of organizations and initiatives around the globe.

Assertions, Protocols and Bindings. The SAML specification [9] is of central importance, as it defines the syntax and semantics of essential SAML structures such as `<AuthnRequest>`, `<Response>` and `<Assertion>`, which are conveyed from the Service Provider (SP) over the User Agent (UA) to the Identity Provider (IdP) and back.

SAML Bindings [8] specify how the structures defined in SAML [9] are bound to different transport protocols such as RFC 2616 [26], SOAP[5] and PAOS [1] for example.

[4]Please note that contrary to common belief, TLS client certificates do not need to be checked against any PKI, and do not carry any identity information. They are only used to anonymously authenticate the browser.

[5]See http://www.oasis-open.org/committees/security.
[6]See http://www.projectliberty.org.
[7]See http://kantarainitiative.org/.
[8]See http://shibboleth.internet2.edu/.

Profiles. SAML Profiles [10] defines how the basic SAML structures, protocols [9] and bindings [8] may be used in different application scenarios. While the standard addresses different use cases, the Single Sign-On (SSO) profiles defined in [10, Section 4] are especially important for the present paper. Currently there are the following profiles specified by OASIS:

- *Web Browser SSO Profile* [10, Section 4.1] – in which all messages are transported using HTTP and hence the UA may be a conventional web browser.
- *Enhanced Client or Proxy (ECP) Profile* [10, Section 4.2] – which assumes that the UA is able to receive PAOS-messages according to [1] and send SOAP-messages according to [5]. This profile is about to be updated in SAML ECP v2.0 [50] in order to support the secure Holder-of-Key- [47] and TLS-Channel binding mechanism [7].
- *Holder-of-Key Web Browser SSO Profile* [39] – is a forthcoming SSO-profile, which uses the so-called "Holder-of-Key" subject confirmation method according to SAML Profiles [10, Section 3.1] and hence a cryptographic binding between the user and her assertion. The main motivation for this profile is the fact that the less secure "Bearer" method, where no key material associated with the token is provided, may not be used for the highest security level according to NIST Electronic Authentication Guideline [44].

Additional standards. SAML Metadata [11] specifies an extensible metadata format for SAML system entities such as the IdP and the SP for example. SAML Authentication Context [38] defines a syntax for the definition of authentication context declarations and an initial list of authentication context classes for use with SAML. In the scope of the present contribution the authentication context "Smartcard" defined in [38, Section 3.4.15] is especially important. SAML Conformance Requirements [43] provides the technical requirements for SAML V2.0 conformance. SAML Glossary [34] defines important terms used in SAML. SAML Security and Privacy Considerations [33] discusses security and privacy issues related to SAML.

2.2 Related Work

The security of SAML v1.0 [32] was analyzed in [29] and the discovered flaws led to additional recommendations in version 2.0 of SAML (cf. [41]). Additional vulnerabilities of the artifact profile have been addressed in [30]. A security analysis for a Liberty-enabled client can be found in [45].

A general treatment of security aspects related to the current version of SAML may be found in SAML Security and Privacy Considerations [33]. A formal security analysis of the Web Browser SSO Profile in SAML 2.0 only appeared recently in [3] and revealed a flaw in the SAML implementation of Google Apps. Other steps towards providing security proofs for browser based protocols can be found in [31, 27]. In [51] it was shown that many SAML implementations were susceptible against signature-wrapping attacks.

Analyzing the list of potential attacks against SAML reveals that many threats are due to the missing cryptographic binding between the SAML messages and the underlying transport protocol. The man-in-the-middle (MitM) attack visualized in [18, Figure 5] exploits this weakness.

As of today, there have been different proposals for binding SAML to the underlying TLS channel in order to safeguard against MitM-attacks:

TLS-Federation. In this approach [6], the SAML assertion is sent inside a short-lived X.509 client certificate. The SAML assertion thus may replace other identification data like distinguished names and the certificate has the same validity period as the SAML assertion.

Strong Locked Same Origin Policy. Here [49], the client is strengthened to make reliable security decisions. This is done by using the servers public key as a basis for decisions of the Same Origin Policy, rather than the insecure Domain Name System. The certificate verification procedure specified in BSI TR-03112-7 [23] in which it is checked that the hash of the X.509 certificate obtained during the TLS handshake is included in the Card-Verifiable-Certificate (CVC) may be seen as a variant of this approach.

SAML 2.0 Holder-of-Key Web Browser SSO Profile. This approach also uses TLS with client authentication, but the client certificate does not transport any authorization information. Instead, the SAML token is bound to the public key contained in this certificate, by including this key in a Holder-of-Key assertion [47, 39]. The security of this approach has independently been analyzed in [28]. In [16] the authors described a methodology regarding the fresh and dynamically generation of self-signed certificates used within the authentication and for the binding to the TLS channel.

TLS-Channel Bindings. Finally, the generic channel binding mechanism sketched in [53] can be applied to TLS [2], Single Sign-On [48] and SAML [7].

2.3 Background on eID-activation

As shown in [19] there are already many European member states, which have introduced eID cards supporting electronic authentication and identification. Many eID cards in the field are equipped with X.509-based certificates, which can be used for TLS client authentication [15]. Because the X.509 certificate is transmitted over an unprotected channel during the TLS-handshake and some certificates even contain sensitive identity attributes, such as the name of the card holder for example, this approach is not optimal from a privacy point of view. Against this background it is no surprise that modern eID cards, such as the German "Neuer Personalausweis" (nPA) for example, do *not* use X.509 certificates and TLS for authentication, but the more privacy friendly Extended Access Control (EAC) protocol [21].

In either case the authentication protocol is performed by an eID-Client (eID-C), which is activated by the User Agent (UA). Typically the UA is a web browser and the eID-activation is performed within a web session. There are different mechanisms for eID-activation[9]:

Push Method. This activation method is specified in [20] and uses an `<object>` tag of type `application/vnd.ecard-client`, which is embedded in a web page and contains the address of the eID-Server (eID-S), which performs the authentication on behalf of the SP. This mechanism is similar to the activation mechanism specified in [37]. The Push Method offers as an advantage less complexity of the protocol flow and compliance according to the SAML standard. However, a suitable browser extension is required, which recognizes the type of the `<object>` and *pushes* the required information to the eID-C.

Pull Method. This method is specified in [23] and uses an embedded link pointing to the following URL `http://localhost:24727/eID-Client?=tcTokenURL=..`, which instructs the eID-C to *pull* the required address information – and the corresponding X.509-based TLS server certificate, which is to be checked against the CVC (Card-Verifiable-Certificate) of the eID-S – from a so called TCToken Service (TS). In comparison to the Push Method, the protocol flow here is more complex and is not SAML-compliant. Though, the method does not require a browser extension.

3. NPA-SPECIFIC SAML PROFILE

In order to provide more security, the German Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik, BSI) created a specific SAML profile for the German eID card (Neuer Personalausweis, nPA) specified in the technical directives [23] and [25].

This is one of the most completely specified governmental eID systems, and therefore a suitable target for a security evaluation. An overview on additional profiles that can be used in connection with governmental eID projects is given in Appendix A.1.

3.1 Description

The authentication process is depicted in Figure 1 and comprises the following steps:

(1) *UA* → *SP*: In the first step the user navigates his UA to the SP and requests a restricted resource.

(2) *SP* → *UA*: Since the user is not authenticated yet, the SP returns a web page with an embedded link `<a href=http://localhost:24727/eID-Client ?tcTokenURL=...>`.

(3) *UA* → *eID-C*: The user clicks on the link and the UA performs a corresponding HTTP GET command.

(4) *eID-C* → *SP*: The eID-C extracts the `tcTokenURL`-parameter, which points to the SP.

[9]See `https://www.openecard.org/en/framework/eid-activation` for corresponding figures, which depict the different methods.

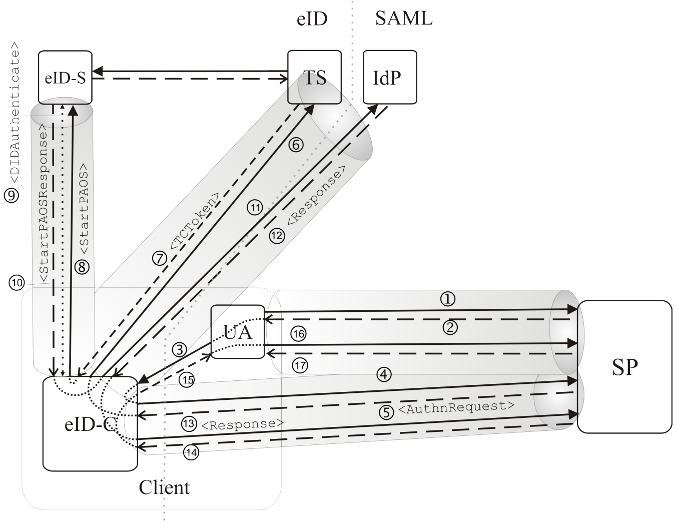

Figure 1: SAML profile according to BSI-TR-03112-7 and BSI-TR-03130 using the Pull Method

(5) *SP* → *eID-C*: The SP in turn initiates the SSO-procedure by returning a URL-encoded `<AuthnRequest>` using the HTTP Redirect Binding [8, Section 3.4].

(6) *eID-C* → *TS*: The eID-C follows the redirect and transports the `<AuthnRequest>` to the TS.

(7) *TS* → *eID-C*: The TS returns a `<TCToken>` to the eID-C containing the connection parameters for the eID-S.

(8) *eID-C* → *eID-S*: As in step (5) of the Web Browser SSO Profile explained in Section A.1.1, the eID-C sends `<StartPAOS>` to the eID-S in order to initiate the authentication procedure.

(9) *eID-S* ↔ *eID-C*: The eID-S and the eID-C perform the eID-specific authentication protocol by exchanging a suitable set of `<DIDAuthenticate>` messages (see [22], [23] and (7) in Section A.1.1).

(10) *eID-S* → *eID-C*: After the eID-S has authenticated the user, the `<StartPAOS Response>` message is returned to the eID-C. Subsequently, the connection is closed (cf. (7) in Section A.1.1).

(11) *eID-C* → *IdP*: The eID-C contacts the IdP to retrieve the result of the authentication procedure.

(12) *IdP* → *eID-C*: The IdP returns a redirect[10] with the URL-encoded `<Response>` to the eID-C.

(13) *eID-C* → *SP*: The `<Response>` is forwarded to the SP.

[10]Please note that the current specification [23, Section 3.6] indeed stipulates that the HTTP Redirect Binding is used for this step (”The SAML Processor returns a redirect to the eService, containing the SAML Authentication Response in HTTP Redirect Binding (see SAML Bindings [8]).”). This is very unfortunate, because this deviation from the Web Browser SSO Profile [10, Section 4.1] will likely cause problems if a lot of attributes are contained in the assertion and prevent that standard SAML SP components can be used in this profile.

(14) $SP \rightarrow eID\text{-}C$: After successful verification of the `<Response>` the SP returns a redirect to the eID-C, which points to the protected resource.

(15) $eID\text{-}C \rightarrow UA$: The eID-C returns the request from step (3) and redirects the UA to the SP.

(16) $UA \rightarrow SP$: The UA follows the redirect and contacts the SP.

(17) $SP \rightarrow UA$: The SP finally returns the protected resource to the UA.

3.2 Technical Analysis

A drawback of this approach is the rather complex communication pattern, which decreases efficiency and makes a complete security analysis nearly impossible.

Interoperability. Interoperability apparently was no major concern of the authors of this standard: SAML standards are modified throughout, such that not a single SAML-conforming SP will be able to connect to this system without major modifications:

- Modifications start with SAML assertions, which must be partially encrypted according to the BSI documents. Partial encryption of SAML assertions is not supported by nearly any current SP implementations, thus, these modified assertions cannot be consumed without major adjustments to the SP.
- Non-standard TLS libraries are required both on server side, and in the eID application, because preshared key TLS needs to be used.
- The HTTP Redirect Binding [8, Section 3.4] in step (13) is not allowed in the Web Browser SSO Profile [10, Section 4.1.2], furthermore encrypted assertions are used, as required by [25, Section 5.8.1], which however is uncommon for the Web Browser SSO profiles and hence most likely not supported by standard SP components.
- The security of the system largely depends on the *Strong Locked Same Origin Policy* (cf. Section 2.2), which is based on the special certificate infrastructure of the BSI system. Other eID systems, which do not utilise a protocol similar to EAC and a compareable PKI concept, can not be used with this SAML profile in a secure manner.

Security. The main security problem of the eID system described above is the fact that the web browser is completely decoupled from the authentication process. This reduces the security level to simple password-based authentication in the network-based attacker model. Additionally, attacks like Session Fixation [40] could exploit vulnerabilities in the session management between user agent and SP. In this manner, all security mechanisms within the authentication could be bypassed. Please note that the following is not an actual attack, but only shows that under the same assumptions that allow password theft from an average internet user, the BSI system can also be compromised. This makes the extra cost involved in setting up such a system questionable.

1. The only step in the protocol flow from Figure 1 that links the authentication process to the web browser is the redirect command which is forwarded to the web browser in message (15). According to the specification, the return to the web session in the browser is done by a refresh URL which "SHOULD be unpredictable and cryptographically strong" [24]. This URL is either the `RefreshAddress` given in the `<TCToken>` in step (7) or a freshly generated URL returned in step (14) by the SP. If this parameter can be observed or influenced by an adversary, a real attack may be possible.

2. In our model for password theft, we assume that the victim is able to verify domain names and to distinguish HTTPS from HTTP connections, but that he is not able to distinguish valid from invalid certificates. Please note that in case of the SP, most probably no Extended Validation Certificates will be used.

3. The adversary, who has full control over (the insecure parts of) the network, may perform a DNS Spoofing attack to lure the victim to a web page under his control. This web page can be TLS protected, but we assume that the victim doesn't care about error messages [52], and thus will manually accept the attacker's certificate as valid and establish an HTTPS connection to this web page.

4. The adversary now establishes a TLS channel to the SP, and requests a protected resource. He simply relays message (2) to the victim, thus triggering the eID authentication.

5. With message (15), the information flow is returned to the browser, and it will automatically perform the requested redirect. Since HTTP redirects are URL based, the target of this redirect will be the web page of the adversary, not the SP.

6. The adversary can now forward this HTTPS request to the SP, thus effectively claiming the victim's authentication and gaining full access to the SP.

Again, please note that the flow above does not describe a real-world attack, but shows one critical point in the whole system (message (15)). Moreover, there is no security gain in employing such a complex and expensive system over simple password-based authentication.

TLS bindings. In the BSI system, none of the TLS bindings specified by OASIS (SAML Holder-of-Key) or IETF (RFC 5929) can be applied, since the TLS channel between browser and SP has no relation at all to the authentication process.

4. HOLDER-OF-KEY WEB BROWSER SSO PROFILE

4.1 Description

A standardized approach to prevent Cross-Site Scripting (XSS) and MitM-attacks against SSO systems is to use the SAML Holder-of-Key (HoK) Profile defined by OASIS [39]. The main idea is to use a TLS client certificate to bind a SAML assertion to a certain browser. Please note that identity information will only be transported in the SAML assertion, thus no PKI is required[11] for the client certificate – a self-signed certificate would be sufficient.

The overall process flow is similar to the procedure used within the Web Browser Single Sign-On Profile with Push Method, which is described in detail in Section A.1.1. In the following only the differences are described:

(3) $UA \rightarrow IdP$: In step 3 the UA forwards the `<AuthnRequest>` to the IdP. Additionally, the UA has to present in this step an X.509 certificate in conjunction with the TLS connection to the IdP. Please note that the certificate does not need to be trusted and does not provide any further information regarding the user. But it has to be presented within the TLS handshake where the possession of the corresponding private key will be confirmed. In this step the following cases can be distinguished:

   (a) The user already has an X.509 certificate and is able to use it in the UA for the TLS handshake with the IdP. In this case it is not necessary to issue another certificate and this certificate can be used for the HoK-binding. If this certificate and the corresponding private key are hosted on the eID-C under consideration and accessed via the PKCS#11 interface of the browser for example, the additional steps (4)-(8) may even be dropped entirely.

   (b) The user does not possess any certificates and thus is not able to fulfill the requirements of the IdP for the TLS channel binding. However, the IdP is able to enforce the generation of a new certificate and automatically import it into the UA. This can be achieved on the fly and without any user interaction by using HTML5 in combination with the `keygen`-Tag[12]. However, since the automated deleting of the generated certificates is not possible, this feature should be used carefully regarding the key storage of the browser. Another option would be that the user creates a self-signed certificate and uses it within the TLS handshake. This self-signed certificate may be bound to a specific service provider (see also [16, 12, 4]).

(10) $IdP \rightarrow UA$: After the successful user authentication, the IdP processes the retrieved information regarding the user and generates and signs the `<Assertion>` [47] which contains the X.509 certificate used by the UA within the TLS connection with the IdP.

(11) $UA \rightarrow SP$: In this step the `<Response>` is delivered by the UA to the SP. As in step 3 the UA presents the same X.509 certificate to the SP in conjunction with the TLS session and confirms the possession of the according private key. Subsequently, the SP verifies the validity of the received authentication token and additionally compares the X.509 certificate from the current TLS session with the certificate included in the `<Response>`. If the `<Response>` passes all tests, the SP grants the access to the requested resources in step (12).

[11]Many people who worked on (failed) projects on PKI infrastructures for TLS client certificates in the past share the belief that client certificates are technically difficult to handle; this is however only the case if PKI validation of such a certificate is involved, which is not the case here.
[12]http://www.w3schools.com/tags/tag_keygen.asp

The main advantage of this profile is that it can be used with arbitrary eID cards and authentication protocols. It is based on an emerging standard [39] (SAML-HoK). The drawback o f th is appro ac h is that it may be necessary to create a certificate on the fly which might decrease efficiency and that using the same certificate at different services may decrease privacy, because the certificate could be used as identifier for the UA. While this problem could in principle be mitigated by using SP-specific certificates [16, 12] (or similar key material [4], if the HoK-specification [47] would be adjusted accordingly), this would either require additional functionality and interfaces in the eID-C for the dynamic handling (creation, usage and deletion) of HoK-specific certificates or decrease usability, as the user would need to delete the dynamically created certificates in a manual process as currently there do not seem to be commonly supported browser APIs for this purpose.

4.2 Optimized version of Holder-of-Key Web Browser SSO Profile

We now show how to optimize the Holder-of-Key Web Browser SSO Profile to allow for efficient and secure authentication using eID-specific methods. Figure 2 depicts a streamlined process, where the SAML-specific components UA and IdP act as proxies for the eID-specific components $eID\text{-}C$ and $eID\text{-}S$, respectively.

The full authentication process involves the following steps:

(1) $UA \rightarrow SP$: The user starts the process by requesting a restricted resource from the SP.

(2) $SP \rightarrow UA$: The SP returns an `<AuthnRequest>` within an *HTTP Redirect* towards the IdP.

(3) $UA \rightarrow IdP$: The UA forwards the `<AuthnRequest>` to the IdP. As stated in Section 4.1, the UA has to present his X.509 certificate (either an existing or a freshly generated one).

(4) $IdP \rightarrow eID\text{-}S$: The IdP starts a new authentication session with the eID-S.

(5) $eID\text{-}S \rightarrow IdP$: The eID-S may provide parameters for the establishment of a secure channel to the IdP. The channel is established afterwards.

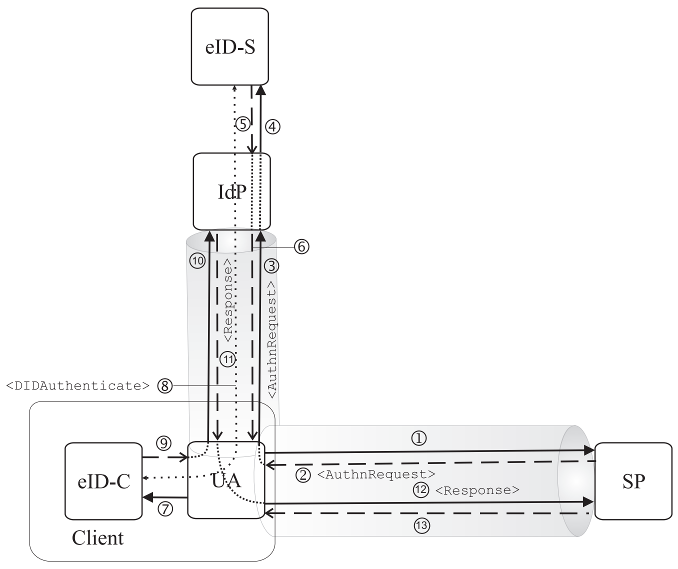

Figure 2: Optimized version of Holder-of-Key Web Browser SSO Profile

(6) $IdP \rightarrow UA$: The IdP forwards the session parameters to the UA and delivers a specific JavaScript that allows the UA to function as proxy towards the IdP.

(7) $UA \rightarrow eID-C$: The UA executes the JavaScript, which in turn activates the eID-C via the localhost-interface according to [24, 3.2.1].

(8) $eID\text{-}C \Leftrightarrow eID\text{-}S$: The eID-C and eID-S perform the eID-specific authentication protocol by exchanging `<StartPAOS>`, a suitable set of `<DIDAuthenticate>` messages and `<StartPAOSResponse>`.

Throughout this communication, the IdP acts as a proxy for the eID-S and forwards the corresponding messages to and from eID-S. In a similar way, the UA forwards all messages to and from the eID-C. Both IdP and UA use their previously established TLS-channel to send the messages through.

(9) $eID\text{-}C \rightarrow UA$: After the authentication between eID-C and eID-S has been performed, eID-C sends a *Redirect* message to the UA, redirecting him to the IdP.

(10) $UA \rightarrow IdP$: The UA follows the redirect to the IdP to request the `<Response>`.

(11) $IdP \rightarrow UA$: The IdP provides the `<Response>` (including the certificate, the UA presented to it in step (3)) to the UA, along with a redirect to the actual SP that requested the authentication.

(12) $UA \rightarrow SP$: The UA forwards the `<Response>` to the SP. The SP now checks, if the `<Response>` is accordingly signed by the IdP and has not been tampered with. The SP also checks, if the certificate included within the `<Response>` matches the one from the actual TLS-channel from the UA to the SP.

(13) $SP \rightarrow UA$: In case both checks from step (12) pass, the SP extracts the authentication information from the `<Response>` and provides the UA with the requested resource.

This optimized variant of the Holder-of-Key Web Browser SSO Profile limits the number of required TLS-channels to two. The UA is the endpoint to both of these channels with the SP, resp. the IdP being the opposite endpoints. By applying the Holder-of-Key Binding the authentication response `<Response>` is cryptographically bound to the TLS-channel between the UA and the SP. Furthermore, the proposed profile combines the advantages of the both Push and Pull methods, see Section 2 and does not require any browser extension.

5. CONCLUSION

The discussion of the different approaches and SAML profiles in the previous sections reveals that there are various possibilities to integrate eID and SAML systems, which have different features with respect to security, efficiency and interoperability.

It is not surprising that the easiest way to integrate eID and SAML is to use the Web Browser SSO Profile [10, Section 4.1] and this approach has the striking charm that eID-based authentication can be used with standard SP-components, which are already available in the field. On the other hand it is well known that using Bearer tokens is not optimal from a security point of view as the assertions could be stolen and misused by a sufficiently strong attacker, which is able to perform a suitable attack.

We showed that in spite of the extensive security measures specified in [23] the nPA-specific SAML profile discussed in Section 3 can be successfully attacked in a similar manner, i.e. using an MitM-attack in the TLS-channel between the UA and the SP.

We also showed that it is possible to significantly enhance the security by moderate changes to the SP and IdP to support the more secure Holder-of-Key-Binding according to [39]. As discussed above, this profile does not require any modifications to the eID-C, but only requires the existence (or on the fly generation) of a certificate in the UA. As this certificate however is transmitted in clear text during the TLS handshake it could be misused as identifier of the user and hence this approach induces additional privacy threats. While it would be possible to create and use SP-specific certificates, the deletion of them from the local certificate store is problematic, because it would either require manual intervention of the user, or additional functionality in the eID-C, such as a PKCS#11 interface for example.

An alternative to prevent MitM-attacks would be to use the SAML-TLS-Channel-Binding mechanism according to [2, 7], but this would currently require that the eID-C is realized as Java Applet. Furthermore, it is an interesting question whether and when browsers support the extraction of key material from a TLS channel as specified in [46] over some standardized API such as the Web Crypto API [13].

6. REFERENCES

[1] Robert Aarts. Liberty Reverse HTTP Binding for SOAP Specification. Liberty Alliance Specification, Version 1.0, 2003. https://www.projectliberty.org/liberty/content/download/2008/13941/file/liberty-paos-v1.0.pdf.

[2] J. Altman, N. Williams, and L. Zhu. Channel bindings for tls. Request For Comments – RFC 5929, July 2010. http://www.ietf.org/rfc/rfc5929.txt.

[3] A. Armando, R. Carbone, L. Compagna, J. Cuellar, and L. Tobarra. Formal analysis of SAML 2.0 web browser single sign-on: breaking the SAML-based single sign-on for google apps. ACM Workshop on Formal Methods in Security Engineering, 2008. http://www.ai-lab.it/armando/pub/fmse9-armando.pdf.

[4] D. Balfanz and R. Hamilton. Transport layer security (tls) channel ids. IETF Internet Draft (draft-balfanz-tls-channelid-00), expires 12.05.2013, 2013. http://tools.ietf.org/html/draft-balfanz-tls-channelid-00.

[5] Don Box, David Ehnebuske, Gopal Kakivaya, Andrew Layman, Noah Mendelsohn, Henrik Frystyk Nielsen, Satish Thatte, and Dave Winer. Simple object access protocol (soap) 1.1. W3C Note: Simple Object Access Protocol (SOAP) 1.1, May 2000. http://www.w3.org/TR/2000/NOTE-SOAP-20000508.

[6] Bud P. Bruegger, Detlef Hühnlein, and Jörg Schwenk. Tls-federation – a secure and relying-party-friendly approach for federated identity management. In *Proceedings of* BIOSIG 2008: Biometrics and Electronic Signatures, volume 137 of *Lecture Notes in Informatics (LNI)*, pages 93–104. GI-Edition, 2008. http://www.ecsec.de/pub/TLS-Federation.pdf.

[7] Scott Cantor. Saml v2.0 channel binding extensions version 1.0. OASIS Working Draft 05, 22.08.2011, 2011. https://www.oasis-open.org/committees/download.php/43302/sstc-saml-channel-binding-ext-v1.0-wd05-diff.pdf.

[8] Scott Cantor, Frederick Hirsch, John Kemp, Rob Philpott, and Eve Maler. Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS Standard, 15.03.2005, 2005. http://docs.oasisopen.org/security/saml/v2.0/saml-bindings-2.0-os.pdf.

[9] Scott Cantor, John Kemp, Rob Philpott, and Eve Maler. Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS Standard, 15.03.2005, 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf.

[10] Scott Cantor, John Kemp, Rob Philpott, and Eve Maler. Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS Standard, 15.03.2005, 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf.

[11] Scott Cantor, Jahan Moreh, Rob Philpott, and Eve Maler. Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS Standard, 15.03.2005, 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf.

[12] Ed. D. Balfanz, D. Smetters, M. Upadhyay, and A. Barth. Tls origin-bound certificates. IETF Internet Draft (draft-balfanz-tls-obc-01), expires 16.05.2012, 2011. http://tools.ietf.org/id/draft-balfanz-tls-obc-01.txt.

[13] David Dahl and Ryan Sleevi. Web Cryptography API. W3C Working Draft 8 January 2013, 2013. http://www.w3.org/TR/WebCryptoAPI/.

[14] C. de Laat, G. Gross, L. Gommans, J. Vollbrecht, and D. Spence. Generic aaa architecture. Request For Comments – RFC 2903, August 2000. http://www.ietf.org/rfc/rfc2903.txt.

[15] T. Dierks and E. Rescorla. The transport layer security (tls) protocol version 1.2. Request For Comments – RFC 5246, August 2008. http://www.ietf.org/rfc/rfc5246.txt.

[16] Michael Dietz, Alexei Czeskis, Dirk Balfanz, and Dan S. Wallach. Origin-bound certificates: a fresh approach to strong client authentication for the web. In *Proceedings of the 21st USENIX conference on Security symposium*, Security'12, pages 16–16. USENIX Association, 2012.

[17] Michael Doujak, Gerhard Hassenstein, Markus Limacher, Marcel Vinzens, Marc Zweiacker, Thomas Moretti, and Urs Bürge. SuisseID Specification – Digital Certificates and Core Infrastructure Services. eCH-0113, Version 1.5, 30.11.2011, 2011. http://tinyurl.com/suisse-id-spec-1-5.

[18] Jan Eichholz, Detlef Hühnlein, and Jörg Schwenk. Samlizing the european citizen card. In *Proceedings of* BIOSIG 2009: Biometrics and Electronic Signatures, volume 155 of *Lecture Notes in Informatics (LNI)*, pages 105–117. GI-Edition, 2009. http://www.ecsec.de/pub/SAMLizing-ECC.pdf.

[19] Anja Lehmann et. al. Survey and Analysis of Existing eID and Credential Systems. FutureID Deliverable, D32.1, Version 0.1, March 2013. https://publicwiki-01.fraunhofer.de/Future_ID/images/0/05/D32.1_v.0.1.pdf.

[20] Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik). eCard-API-Framework – protocols. Technical Directive (BSI-TR-03112), Version 1.1.1, Part 7, 2011. http://docs.ecsec.de/BSI-TR-03112-7-v1.1.1.

[21] Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik). Advanced Security Mechanism for Machine Readable Travel Documents - Extended Access Control (EAC), Password Authenticated Connection Establishment (PACE), and Restricted Identification (RI). Technical Directive (BSI-TR-03110), Version 2.10, 2012. http://docs.ecsec.de/BSI-TR-03110.

[22] Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik). eCard-API-Framework – iso24727-3-interface. Technical Directive (BSI-TR-03112), Version 1.1.2, Part 4, 2012. http://docs.ecsec.de/BSI-TR-03112-4.

[23] Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik). eCard-API-Framework – protocols. Technical Directive (BSI-TR-03112), Version 1.1.2, Part 7, 2012. http://docs.ecsec.de/BSI-TR-03112-7.

[24] Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik). eCard-API-Framework – protocols. Technical Directive (BSI-TR-03112), Version 1.1.2, Part 7, 2012. http://docs.ecsec.de/BSI-TR-03112-7.

[25] Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik). eID-Server. Technical Directive (BSI-TR-031030), Version 1.6, 20.04.2012, 2012. http://docs.ecsec.de/BSI-TR-03130.

[26] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol – http/1.1. Request For Comments – RFC 2616, Juni 1999. http://www.ietf.org/rfc/rfc2616.txt.

[27] Sebastian Gajek. A universally composable framework for the analysis of browser-based security protocols. In Joonsang Baek, Feng Bao, Kefei Chen, and Xuejia Lai, editors, *Provable Security – Second International Conference, ProvSec 2008, Shanghai, China, October 30 - November 1*, volume 5324 of *Lecture Notes in Computer Science*, pages 283–297. Springer, 2008.

[28] Sebastian Gajek, Tibor Jager, Mark Manulis, and Jörg Schwenk. A browser-based kerberos authentication scheme. In Sushil Jajodia and Javier López, editors, *Computer Security - ESORICS 2008, 13th European Symposium on Research in Computer Security, Málaga, Spain, October 6-8, 2008. Proceedings*, volume 5283 of *Lecture Notes in Computer Science*, pages 115–129. Springer, August 2008.

[29] Thomas Groß. Security analysis of the saml single sign-on browser artifact profile. In *Annual Computer Security Applications Conference, December 8-12, 2003, Aladdin Resort & Casino Las Vegas, Nevada, USA*, 2003. http://www.acsac.org/2003/papers/73.pdf.

[30] Thomas Groß and Birgit Pfitzmann. SAML artifact information flow revisited. In *In IEEE Workshop on Web Services Security (WSSS)*, pages 84–100, Berkeley, May 2006. IEEE. http://www.zurich.ibm.com/security/publications/2006/GrPf06.SAML-Artifacts.rz3643.pdf.

[31] Thomas Groß, Birgit Pfitzmann, and Ahmad-Reza Sadeghi. Browser model for security analysis of browser-based protocols. In *ESORICS: 10th European Symposium on Research in Computer Security*, volume 3679 of *Lecture Notes in Computer Science*, pages 489–508, Berlin, Germany, 9 2005. Springer–Verlag. http://eprint.iacr.org/2005/127.pdf.

[32] Phillip Hallam-Baker and Eve Maler. Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML). OASIS Standard, 05.11.2002, 2002. http://www.oasis-open.org/committees/download.php/2290/oasis-sstc-saml-1.0.zip.

[33] Frederick Hirsch, Rob Philpott, and Eve Maler. Security and Privacy Considerations for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS Standard, 15.03.2005, 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-sec-consider-2.0-os.pdf.

[34] Jeff Hodges, Rob Philpott, and Eve Maler. Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS Standard, 15.03.2005, 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-glossary-2.0-os.pdf.

[35] Oracle Inc. Java-to-JavaScript Communication. http://docs.oracle.com/javase/7/docs/technotes/guides/plugin/developer_guide/java_js.html.

[36] Java.net. LiveConnect Support in the New Java Plug-In Technology. http://jdk6.java.net/plugin2/liveconnect/.

[37] Michael B. Jones and Michael McIntosh. Identity Metasystem Interoperability Version 1.0. OASIS Standard, July 2009. http://docs.oasis-open.org/imi/identity/v1.0/os/identity-1.0-spec-os.pdf.

[38] John Kemp, Scott Cantor, Prateek Mishra, Rob Philpott, and Eve Maler. Authentication Context for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS Standard, 15.03.2005, 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf.

[39] N. Klingenstein. SAML V2.0 Holder-of-Key Web Browser SSO Profile. OASIS Committee Specification 02, 10.08.2010, 2010. http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-holder-of-key-browser-sso.pdf.

[40] Mitja Kolše. Session Fixation Vulnerability in Web-based Applications. Technical report, ACROS Securit, 2002.

[41] John Linn and Prateek Mishra. SSTC Response to 'Security Analysis of the SAML Single Sign-on Browser - Artifact Profile'. OASIS Working Draft 01, 24.01.2005, 2005. http://www.oasis-open.org/committees/download.php/11191/sstc-gross-sec-analysis-response-01.pdf.

[42] Eve Maler, Prateek Mishra, and Rob Philpott. Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML) V1.1. OASIS Standard, 02.09.2003, 2003. http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf.

[43] Prateek Mishra, Rob Philpott, and Eve Maler. Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS Standard, 15.03.2005, 2005. http://docs.oasis-open.org/security/saml/v2.0/saml-conformance-2.0-os.pdf.

[44] National Institute of Standards and Technology. Electronic Authentication Guideline. NIST Special Publication 800-63 Version 1.0.2. http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63V1_0_2.pdf.

[45] Birgit Pfitzmann and Michael Waidner. Analysis of liberty single-sign-on with enabled clients. *Internet Computing, IEEE*, 7(6):38–44, 2003. http://www.cs.ru.nl/~jhh/pub/secsem/pfitzmann2003liberty-single-sign-on.pdf.

[46] E. Rescorla. Keying material exporters for transport layer security (tls). Request For Comments – RFC 5705, March 2010. http://www.ietf.org/rfc/rfc5705.txt.

[47] Tom Scavo. SAML V2.0 Holder-of-Key Assertion Profile. OASIS Committee Specification 02, 23.01.2010, 2010.

http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml2-holder-of-key-cs-02.pdf.

[48] Jörg Schwenk, Florian Kohlar, and Marcus Amon. The power of recognition: secure single sign-on using tls channel bindings. In *Proceedings of the 7th ACM workshop on Digital identity management*, DIM '11, pages 63–72, New York, NY, USA, 2011. ACM.

[49] Jörg Schwenk, Lijun Liao, and Sebastan Gajek. Stronger bindings for saml assertions and saml artifacts. In *Proceedings of the 5th ACM CCS Workshop on Secure Web Services (SWS'08)*, pages 11–20. ACM Press, 2008.

[50] Scott Cantor et al. SAML V2.0 Enhanced Client or Proxy Profile Version 2.0. OASIS Working Draft 07, 08.04.2013, 2013. https://www.oasis-open.org/committees/download.php/48788/sstc-saml-ecp-v2.0-wd07.pdf.

[51] Juraj Somorovsky, Andreas Mayer, Jörg Schwenk, Marco Kampmann, and Meiko Jensen. On Breaking SAML: Be Whoever You Want to Be. Proceedings of the 21st USENIX Security Symposium, 2012. http://www.nds.rub.de/media/nds/veroeffentlichungen/2012/08/22/BreakingSAML_3.pdf.

[52] Joshua Sunshine, Serge Egelman, Hazim Almuhimedi, Neha Atri, and Lorrie Faith Cranor. Crying Wolf: An Empirical Study of SSL Warning Effectiveness. In *USENIX Security Symposium*, pages 399–416, 2009.

[53] N. Williams. On the use of channel bindings to secure channels. Request For Comments – RFC 5056, November 2007. http://www.ietf.org/rfc/rfc5056.txt.

APPENDIX

A. SAML PROFILES FOR EID-BASED AUTHENTICATION AND IDENTIFICATION

A.1 Web Browser SSO Profiles

The following SAML profiles are most relevant for the integration of eID-based authentication and identification:

- Web Browser SSO Profile with Bearer Token according to [10, Section 4.1] (see Section A.1.1 and Section A.1.2))
- nPA-specific SAML Profile according to [24, 21] (see Section 3)
- Holder-of-Key Web Browser SSO Profile according to [39] (see Section 4 and Section 4.2)
- eID-Applet with Enhanced Client and Proxy Profile according to [50] (see Section A.2).

A.1.1 Web Browser SSO Profile with Push Method

The most natural and straight forward way to integrate eID and SAML is to use the Web Browser SSO Profile as specified in [10, Section 4.1] together with the Push Method for eID-activation as specified in [20].

The authentication process is depicted in Figure 3 and comprises the following steps:

(1) $UA \rightarrow SP$: In the first step the user navigates his UA to the SP and requests a restricted resource.

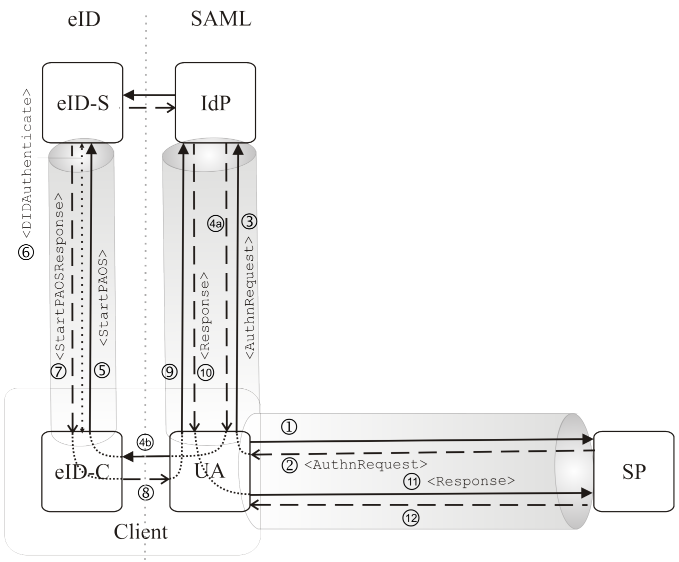

Figure 3: Web Browser SSO Profile with Push Method

(2) $SP \rightarrow UA$: Since the user is not authenticated yet, the SP starts the SSO procedure and issues an `<AuthnRequest>`, which will be sent to the IdP via the UA.

(3) $UA \rightarrow IdP$: The UA forwards the `<AuthnRequest>` to the IdP.

(4) $IdP \rightarrow eID\text{-}C$: The IdP performs the eID-activation by embedding an `<object` into the returned web page (4a), which is pushed to the eID-C (4b).

(5) $eID\text{-}C :\rightarrow eID\text{-}S$: The eID-C sends `<StartPAOS>` to the eID-S in order to initiate the authentication procedure.

(6) $eID\text{-}S :\leftrightarrow eID\text{-}C$: The eID-S and the eID-C perform the eID-specific authentication protocol by exchanging a suitable set of `<DIDAuthenticate>` messages (see [22] and [23].

(7) $eID\text{-}S :\rightarrow eID\text{-}C$: After the eID-S has authenticated the user, the `<StartPAOS Response>` message is returned to the eID-C. Subsequently, the connection is closed.

(8) $eID\text{-}C :\rightarrow UA$: The eID-C redirects the UA to the IdP.

(9) $UA \rightarrow IdP$: The UA follows the redirect and contacts the IdP.

(10) $IdP \rightarrow UA$: After the IdP is informed about the performed authentication it creates an `<Assertion>`, which is embedded in a corresponding `<Response>`, which is returned to the UA.

(11) $UA \rightarrow SP$: The `<Response>` is sent to the SP.

(12) $SP \rightarrow UA$: After successful verification of the `<Response>` the SP returns the protected resource.

The most important advantage of this profile is that it uses standard SAML components, which are already deployed in practice. Furthermore the `<Assertion>` contained in the `<Response>` (see step (10)-(11)) is only a Bearer Token, which may be stolen and misused by a sufficiently strong attacker, which is able to mount a Cross-Site-Scripting- (XSS) or MitM-attack for example.

A.1.2 Web Browser SSO Profile with Pull Method

As shown in Figure 4 a similar authentication process can be implemented with the Pull Method.

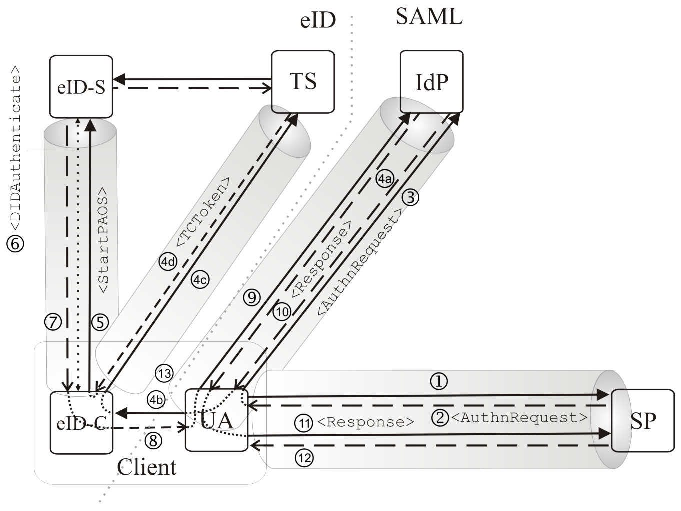

Figure 4: Web Browser SSO Profile with Pull Method

While the process in steps (1)-(3) and (5)-(12) is identical to the description above, step (4) is more changed as follows:

(4a) *IdP → UA*: In the first step a web page with an embedded link [13] is returned.

(4b) *UA → eID-C*: This link is activated in step (4b) by JavaScript or manual intervention of the user.

(4c) *eID-C → TS*: The eID-C contacts the TS with an HTTP GET.

(4d) *TS → eID-C*: Finally the `<TCToken>`-structure is returned, which contains the address of the eID-S and other information.

The security characteristics of this approach are identical to the profile above. The major advantage is that the UA does not need to be equipped with a browser extension. On the other side, step (4b) requires JavaScript or manual intervention, which would reduce usability. Furthermore this profile is slightly less efficient, as it requires the additional steps (4c) and (4d) to retrieve the `<TCToken>`.

A.2 eID-Applet with Holder-of-Key Web Browser SSO or ECP Version 2.0 Profile

Alternatively, a signed eID-Applet (see Figure 5) could be used to generate self-signed certificates if required and establish the HoK-specific TLS-sessions itself or even implement the Enhanced Client and Proxy Profile [50] using the TLS-channel binding according to [2, 7].

Another option, which is similar to the CORS-specific solution above, would be to trigger the establishment of HoK-specific TLS-sessions in the UA via asynchronous JavaScript-functions, which are called via [36] and [35] communication,

[13] `<a href=http://localhost:24727/eID-Client?=tcTokenURL=.../>`

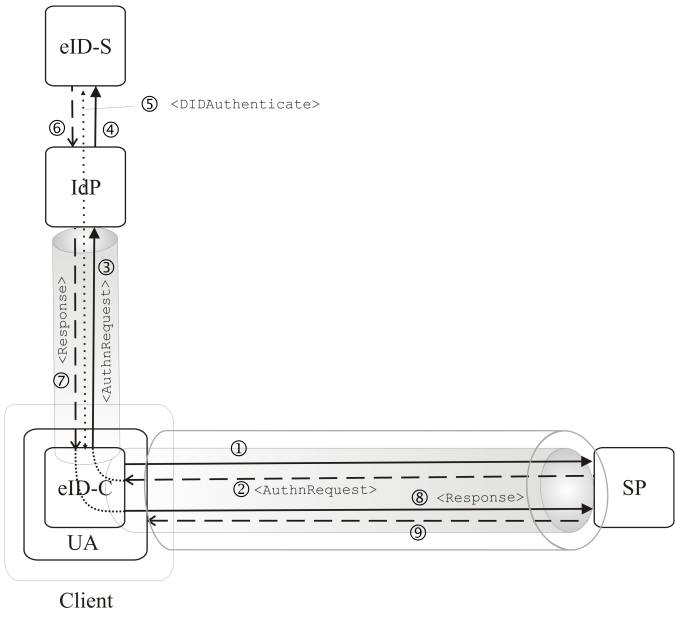

Figure 5: eID-Applet with Holder-of-Key Web Browser SSO or ECP Version 2.0 Profile

which is possible if the applet is started with the respective configuration parameter (`MAYSCRIPT`).