SAMLizing the European Citizen Card

Jan Eichholz[1], Detlef Hühnlein[2], Jörg Schwenk[3]

[1] Giesecke & Devrient GmbH, Prinzregentenstraße 159, 81677 München,
jan.eichholz@gi-de.com

[2] secunet Security Networks AG, Sudetenstraße 16, 96247 Michelau,
detlef.huehnlein@secunet.com

[3] Ruhr Universität Bochum, Universitätsstr. 150, 44780 Bochum
joerg.schwenk@rub.de

Abstract: While the use of Federated Identity Management and Single Sign-On based on the Security Assertion Markup Language (SAML) standards becomes more and more important, there are quite a few European countries which are about to introduce national ID cards, which are compliant to the European Citizen Card (ECC) specification prCEN 15480. The present contribution shows how these two seemingly opposite approaches may be integrated in a seamless and secure fashion such that it is possible to use the security features of the ECC in a federated scenario, which allows easy integration of Service Providers.

1 Introduction

In the area of Identity Management there seem to be two major trends at the moment, which are addressed in the EU funded project STORK[1]: On the one hand side, Federated Identity Management solutions are increasingly used in practice as they allow to implement Single Sign-On 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. On the other side quite a few European countries are about to introduce national ID cards, which are compliant to the European Citizen Card specification [CEN15480-1, CEN15480-2, CEN15480-3, CEN15480-4]. Hence it is natural to investigate how both approaches can be integrated such that systems which aim at implementing the eService directive [2006/123/EC] may combine the security of the ECC with the easy integration of Service Providers in SAML.

The rest of the paper is structured as follows: Section 2 provides the necessary background on the Security Assertion Markup language and the European Citizen Card supporting the Extended Access Control (EAC) protocol [BSI-TR-03110(V2.01)] and briefly considers related work. Section 3 explains how the ECC may be "SAMLized" and how an ECC-specific SAML-profile may look like, which may be used as starting point for the development of further specifications in STORK and standardization in CEN TC 224 WG 15 and/or OASIS Security TC.

[1] See www.eid-stork.eu

2 Background on SAML and the European Citizen Card

This section contains background information, which is helpful to understand the main contribution in Section 3. While Section 2.1 recalls the main aspects of the Security Assertion Markup Language (SAML), Section 2.2 provides some basic information about the European Citizen Card (ECC) specifications.

2.1 Background on SAML-based Single Sign-On

In this section the necessary background on the Security Assertion Markup Language (SAML) and in particular SAML-based Single Sign-On is provided.

SAML is a family of standards, which has been developed by the OASIS Security Services Technical Committee[2] 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 [SAML(v1.0)], v1.1 [SAML(v1.1)] and v2.0 [SAML(v2.0)]) have been influenced by previous work at IETF [RFC2903] and projects like the Liberty Alliance[3] and Shibboleth[4].

2.1.1 Overview of SAML Version 2.0

The current version of SAML consists of the following parts:

- *Assertions and Protocols* [SAML(v2.0)] – is of central importance, as it defines the syntax and semantics of essential SAML-structures such as `<Assertion>`, `<AuthnRequest>` and `<Response>`.
- *Bindings* [SAML-Bind(v2.0)] – specifies how the structures defined in [SAML(v2.0)] are bound to the different transport protocols such as [RFC2616, SOAP(v1.1)] and [PAOS(v1.0)] for example.
- *Profiles* [SAML-Prof(v2.0)] – defines how the basic structures, protocols [SAML(v2.0)] and bindings [SAML-Bind(v2.0)] may be used for different application scenarios. While the standard addresses different use cases, we are particulary interested in the Single Sign-On (SSO) profiles defined in [SAML-Prof(v2.0), Section 4] and therefore will provide more details below.
- *Metadata* [SAML-Meta(v2.0)] – specifies an extensible metadata format for SAML system entities such as the Identity Provider and the Service Provider (cf. Figure 1) for example.

[2]See `http://www.oasis-open.org/committees/security`.
[3]See `http://www.projectliberty.org`.
[4]See `http://shibboleth.internet2.edu/`.

- *Authentication Context* [SAML-Auth(v2.0)] – 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 [SAML-Auth(v2.0), Section 3.4.15] is especially important.
- *Conformance Requirements* [SAML-Conf(v2.0)] – provides the technical requirements for SAML V2.0 conformance.
- *Glossary* [SAML-Glos(v2.0)] – defines important terms used in SAML.
- *Security and Privacy Considerations* [SAML-SecP(v2.0)] – discusses security and privacy issues related to SAML. Please refer to Section 2.1.3 for more information on security aspects related to SAML.

2.1.2 SAML-based Single Sign-On (SSO)

As in the generic SSO-scenario introduced in [BHS08, Section 2.1] there is a User (U) with User Agent (UA), who wants to access the services offered by the Service Provider (SP). But as the User is not able to authenticate at the SP directly it needs to contact the Identity Provider (IP) in order to be authenticated and equipped with a SAML-assertion, which may finally be consumed by the SP.

Currently the following SSO-profiles are (about to be) standardized in SAML:

- *Web Browser SSO Profile* [SAML-Prof(v2.0), Section 4.1] – in which all messages are transported using plain http and hence the UA may be a conventional web-browser.
- *Enhanced Client or Proxy (ECP) Profile* [SAML-Prof(v2.0), Section 4.2] – which assumes that the UA is able to receive PAOS-messages according to [PAOS(v1.0)] and send SOAP-messages according to [SOAP(v1.1)].
- *Holder-of-Key Web Browser SSO Profile* [SAML-HoKWebSSO] – is a forthcoming SSO-profile, which – unlike the two profiles above which only support the less secure ”Bearer” subject confirmation method according to [SAML-Prof(v2.0), Section 3.3] – uses the so called ”Holder of Key” subject confirmation method according to [SAML-Prof(v2.0), Section 3.1] and hence a cryptographic binding between the User and her assertion. A main motivation for this profile is the fact that the less secure ”Bearer” methods may not be used for for the higher security levels according to [NIST-800-63].

Web Browser SSO Profile. For the Web Browser SSO Profile [SAML-Prof(v2.0), Section 4.1] the following basic steps (cf. Figure 1) are performed:

1. $UA \rightarrow SP$: The User Agent contacts the Service Provider by sending some HTTP-request, such as GET for example.

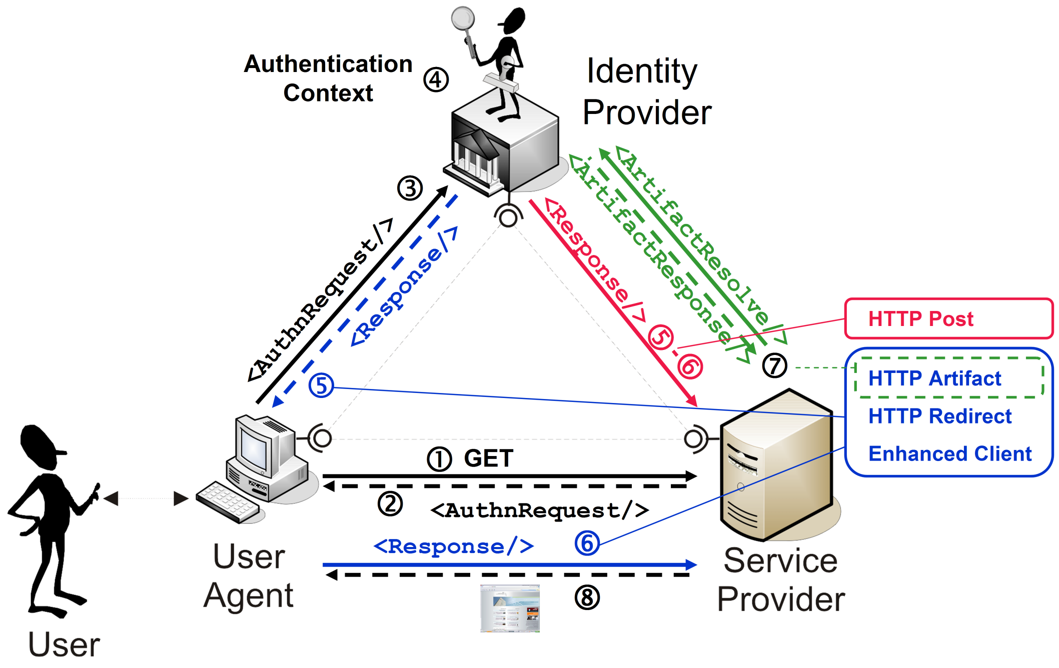


Figure 1: Single Sign-On based on SAML

2. $SP \rightarrow UA$: The Service Provider answers with a HTTP-response that contains an `<AuthnRequest>`-element (cf. [SAML(v2.0), Section 3.4.1]). The details of the encoding of this element depend on the binding and there are the following options:

   - HTTP Redirect Binding [SAML-Bind(v2.0), Section 3.4]
   - HTTP POST Binding [SAML-Bind(v2.0), Section 3.5]
   - HTTP Artifact Binding [SAML-Bind(v2.0), Section 3.6]

   In this case the Service Provider needs to decide which Identity Provider shall be used in the next step. For this purpose the Identity Provider Discovery Profile defined in [SAML-Prof(v2.0), Section 4.3], which uses a common domain cookie, may be used.

3. $UA \rightarrow IP$: The User Agent sends the `<AuthnRequest>`-element to the Identity Provider using a simple HTTP GET or POST request.

4. IP: In absence of a previously established authenticated session, the Identity Provider authenticates the User according to the requirements of the Service Provider, which may be specified within the `<RequestedAuthentication Context>`-element

inside the `<AuthnRequest>`-element using the predefined authentication classes defined in [SAML-Auth(v2.0)].

5. $IP \rightarrow UA$: Upon successful authentication, the Identity Provider returns a Credential to the User Agent. The type of the returned data again depends on the binding[5]:
   - HTTP Artifact Binding
     In this case the Identity Provider returns a SAML-artifact as defined in Section 3.6.4 of [SAML-Bind(v2.0)].
   - HTTP POST Binding
     In this case the Identity Provider directly sends a `<Response>`-element to the Service Provider, which may contain one or more `<Assertion>`-elements (see [SAML-Prof(v2.0), Section 4.1.4.2] for details).
6. $UA \rightarrow SP$: In case of the HTTP Artifact Binding the User Agent sends the received SAML artifact to the Service Provider.
7. SP: In this step, the Service Provider needs to validate the received Credential. If the HTTP Artifact Binding has been used it will first use the artifact to obtain the corresponding `<Assertion>` from the Identity Provider using the Artifact Resolution Protocol defined in [SAML(v2.0), Section 3.5]. In any case it will then verify the signature contained in the `<Assertion>` which in turn may require the validation of a chain of certificates.
8. $SP \rightarrow UA$: On success, the Service Provider finally returns an HTTP-response that contains the requested resource.

Enhanced Client or Proxy (ECP) Profile. For this profile, which is specified in Section 4.2 of [SAML-Prof(v2.0)], the User Agent needs to support additional functionality. In particular it is necessary in this profile that the User Agent natively supports SOAP [SOAP(v1.1)] and PAOS [PAOS(v1.0)], which is not the case for typical web browsers yet.

For this profile the following basic steps (cf. Figure 1) are performed:

1. $UA \rightarrow SP$: As above the User Agent contacts the Service Provider by sending some HTTP-request and indicates ECP-support by a specific HTTP-header (cf. [SAML-Prof(v2.0), Section 4.2.3.1] for details).
2. $SP \rightarrow UA$: The Service Provider answers with an `<AuthnRequest>`-element (cf. [SAML(v2.0), Section 3.4.1], which is transported using the Reverse SOAP (PAOS) Binding [SAML-Bind(v2.0), Section 3.3], which is in turn based on [PAOS(v1.0)]. Within the `<AuthnRequest>`-element there may be an `<IDPList>`-element, which specifies which Identity Providers are supported by the Service Provider and the enhanced User Agent may involve the User to select the Identity Provider to be contacted in the next step.

[5]Note that the HTTP Redirect Binding may *not* be used in this step, because the `<Response>`-element would typically be be too large to be encoded as URL-parameter.

3. $UA \rightarrow IP$: The User Agent sends the `<AuthnRequest>`-element to the Identity Provider using the SAML SOAP Binding defined in [SAML-Bind(v2.0), Section 3.2], which is in turn based on [SOAP(v1.1)].

4. IP: As above the User is authenticated, if this has not happened before.

5. $IP \rightarrow UA$: Upon successful authentication, the Identity Provider returns a `<Response>`-element, which contains one or more `<Assertion>`-elements, to the User Agent using the SOAP Binding.

6. $UA \rightarrow SP$: The User Agent sends the received `<Response>`-element to the Service Provider using the PAOS Binding.

7. SP: In this step, the Service Provider needs to validate the received Credential and in particular verifies the signature of the `<Assertion>`.

8. $SP \rightarrow UA$: On success, the Service Provider finally returns an HTTP-response that contains the requested resource.

Holder-of-Key Web Browser SSO Profile. In this profile [SAML-HoKWebSSO], which is very similar to the ordinary Web Browser SSO-profile outlined above, it is assumed that the User Agent has an X.509 certificate [X.509:05], which is presented to the Identity Provider in a TLS-handshake [RFC5246] and subsequently bound to the issued `<Assertion>` as explained below (cf. [SAML-HoKAP]). In a similar manner the User Agent is able to establish a TLS-channel with the Service Provider using this X.509 certificate and hence the Service Provider may verify that the presented `<Assertion>` indeed belongs to the User Agent, who holds the private key corresponding to the certificate, and hence the threat that somebody may steal a bearer token (cf. [SAML-SecP(v2.0), Section 7.1.1.3]) is effectively removed.

As defined in [SAML-HoKAP] a holder-of-key SAML assertion, is an assertion containing a `<saml:SubjectConfirmation>` element (cf. [SAML(v2.0), Section 2.4.1.1]), whose `Method` attribute is set to `urn:oasis:names:tc:SAML:2.0:cm:holder-of-key` and contains a `<saml:SubjectConfirmationData>`-element, which in turn contains (a reference to) the X.509 certificate of the User Agent in form of a `<ds:KeyInfo>`-element according to [XML-DSig].

2.1.3 Security aspects of SAML-based Single Sign-On

The security of [SAML(v1.0)] was analyzed in [Gros03] and the discovered flaws lead to additional recommendations in version 2.0 of SAML (cf. [SAML-Resp]). Additional vulnerabilities of the artifact profile have been addressed in [GrPf06]. A security analysis for a Liberty-enabled client can be found in [PfWa03].

A general treatment of security aspects related to the current version of SAML may be found in [SAML-SecP(v2.0)] and analyzing the list of potential attacks reveals that many threats are due to the missing cryptographic binding between the SAML assertions and the underlying transport protocol. Therefore there have recently been proposals for stronger bindings for SAML assertions and artifacts in [GLS2008, BHS08] and a standard profile which features a cryptographic binding is on its way (see above and [SAML-HoKWebSSO]).

A formal security analysis of the browser-based Single Sign-On in SAML 2.0 only appeared recently in [ACC+08] 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 [GPS05, Gaje08].

2.2 European Citizen Card

The CEN technical standard series prTS 15480 [CEN15480-1, CEN15480-2, CEN15480-3, CEN15480-4] describes services, command sets and application contexts of the European Citizen Card (ECC). The appendix of [CEN15480-4] contains profiles for sector specific applications e.g. for electronic health and/or eID cards. The Extended Access Control protocol, used in the eID profile for device authentication and session key agreement between the ECC and its external partner (local and/or via Internet) is derived from [BSI-TR-03110(V2.01)]. This specification extends the previous version of the EAC protocol [BSI-TR-03110(V1.11)], which is already in use for the second generation of electronic passports.

Both protocols belong to the so called modular Extended Access protocol family (mEAC), which design is described in the basic standard for ESIGN cards [CEN14890-1, CEN14890-2]. The mEAC family comprises the following basic protocol components:

- *Password Authenticated Connection Establishment (PACE)* (cf. [BSI-TR-03110(V2.01), Section 4.2])
  which uses a short password to establish a secure channel between the terminal and the card. This protocol is typically used to protect the communication between a local terminal and a contactless card. Please refer to [BFK09] for security aspects of this protocol.
- *Terminal Authentication* (cf. [BSI-TR-03110(V2.01), Section 4.4])
  is a protocol in which the terminal signs a challenge provided by the card in order to be authenticated. Within this protocol the terminal presents a certificate chain to the card, which in particular specifies the authorization of the terminal.
- *Chip Authentication* (see Figure 2, cf. [BSI-TR-03110(V2.01), Section 4.3])
  is a protocol in which the terminal and the card use the Diffie-Hellman primitive to agree on a session key K, which is subsequently used for authentication purposes. For this purpose the domain parameters $\mathcal{D}$ and the static public key PK_{ECC} of the ECC are transmitted to the terminal. The terminal returns an ephemeral public key $\tilde{P}K_{PCD}$, which is used to agree on a common key K. This key is used to derive keys K_{Enc} and K_{MAC} for secure messaging purposes and for the generation of the

authentication token T, which is a computed by applying a message authentication code to the ephemeral public key of the terminal $\tilde{P}K_{PCD}$. Since ephemeral terminal keys are used, the Chip Authentication protocol offers the possibility to bind the EAC session information to the eService (see Section 3.4).

- *Passive Authentication* (cf. [BSI-TR-03110(V2.01), Section 1.1 and Annex A.1.2]) means that sensitive data (e.g. the public key of the card used in the Chip Authentication protocol) are protected by a digital signature according to [RFC3369], which is produced by the so called Document Signer.
- *Restricted Identification* (cf. [BSI-TR-03110(V2.01), Section 4.5]) is a protocol in which the terminal and the card perform a Diffie-Hellman like protocol in order to produce a sector specific pseudonym of the card.

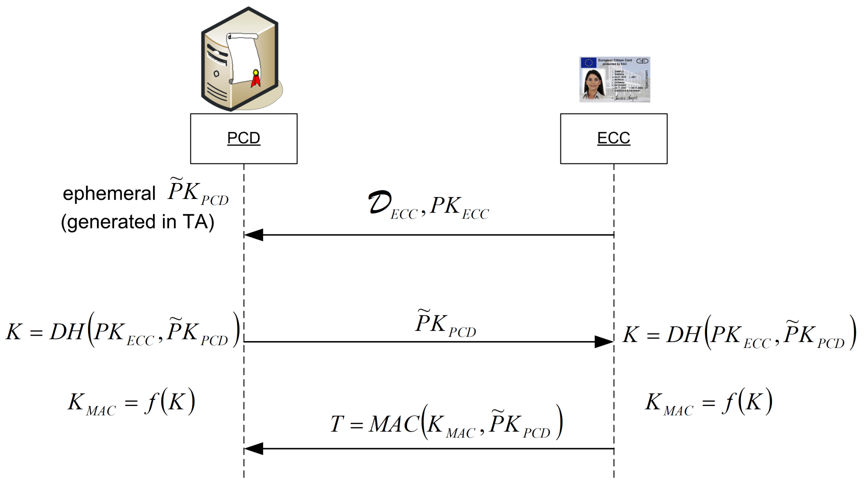

PCD
ECC
ephemeral $\tilde{P}K_{PCD}$ (generated in TA)
$\mathcal{D}_{ECC}, PK_{ECC}$
$K = DH(PK_{ECC}, \tilde{P}K_{PCD})$
$\tilde{P}K_{PCD}$
$K = DH(PK_{ECC}, \tilde{P}K_{PCD})$
$K_{MAC} = f(K)$
$T = MAC(K_{MAC}, \tilde{P}K_{PCD})$
$K_{MAC} = f(K)$

Figure 2: Chip Authentication (Version 2)

Within part 3 of CEN prTS 15480 [CEN15480-3] the application interface for the communication with European Citizen Cards is described. This standard is based on the international smart card standard ISO/IEC 24727 [ISO24727-1, ISO24727-2, ISO24727-3, ISO24727-4] and extends it to cover the specific requirements in the context of eID. The main goal of prTS 15480-3 [CEN15480-3] is to offer a simple web service based interface to the service application at the Service Provider, which allows a simple and card independent access to smart cards. For this purpose there is a distributed eID-middleware architecture (cf. Figure 3), but there is no need for a specific Identity Provider, because the Service Provider simply uses its eID-middleware to access the ECC and authenticate the User. For this purpose (cf. Terminal Authentication above) the Service Provider needs to present a "Card-Verifiable Certificate" according to [ISO7816-8, Annex A.4] to the ECC, which in particular contains information about the access rights of the Service Provider on the ECC.

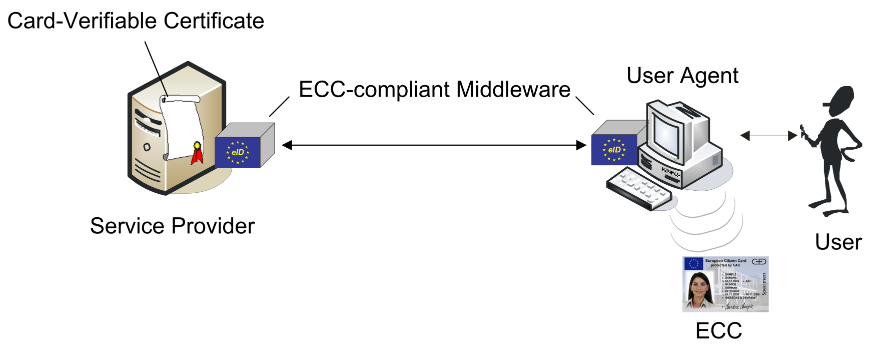


Figure 3: System Architecture according to CEN prTS 15480-3

3 Secure Integration of the ECC into a SAML-environment

In order to implement the eService-Directive [2006/123/EC] it is necessary that European citizen are able to use their national identification token (e.g. an ECC-compliant ID-card) to authenticate at some eService in another EU Member State. As long as not all eServices across Europe directly support the ECC-compliant authentication protocols, such as EAC for example, the use of Federated Identity Management techniques, e.g. based on SAML, may ease the integration of eServices and hence facilitate the implementation of the eService-Directive. On the other side it is necessary to seriously analyze security aspects of such a construction, as a naive integration of a highly secure national ID-card into a SAML-environment may considerably degrade the overall security.

As explained in Section 3.1 there are three main approaches for the secure integration of the European Citizen Card into a SAML-environment. First SAML may be bound to the involved TLS-sessions (cf. Section 3.2). Second the two TLS-sessions may be bound together and may be bound to the EAC-session (cf. Section 3.3). Third it is possible to bind the SAML-Assertion directly to the EAC-protocol (cf. Section 3.4). Finally, the pros and cons as well as the possible combination of these approaches are discussed in Section 3.5.

3.1 Overview, requirements and threats

In order to allow Users, which are equipped with EAC-based eID tokens, to access the services of a Service Provider (SP), which only supports SAML, it is a straightforward approach to make use of a specific eID-Server, which supports both EAC and SAML and may serve as Identity Provider (IP), which ”translates” an EAC-based authentication context into an appropriate SAML-Assertion, which may be consumed by the Service Provider.

As can be seen by comparing Figure 4 and Figure 1 the applied protocol is very similar to the SAML-protocol for an enhanced client, which is capable of performing an EAC-based authentication in step 4. Furthermore it can be seen in Figure 4 that besides the EAC-channel between the ECC and the eID-Server there may be two TLS-channels (TLS_{UA-SP} between the User Agent and the Service Provider (eService) and TLS_{UA-IP} between the User Agent and the Identity Provider (eID-Server)).

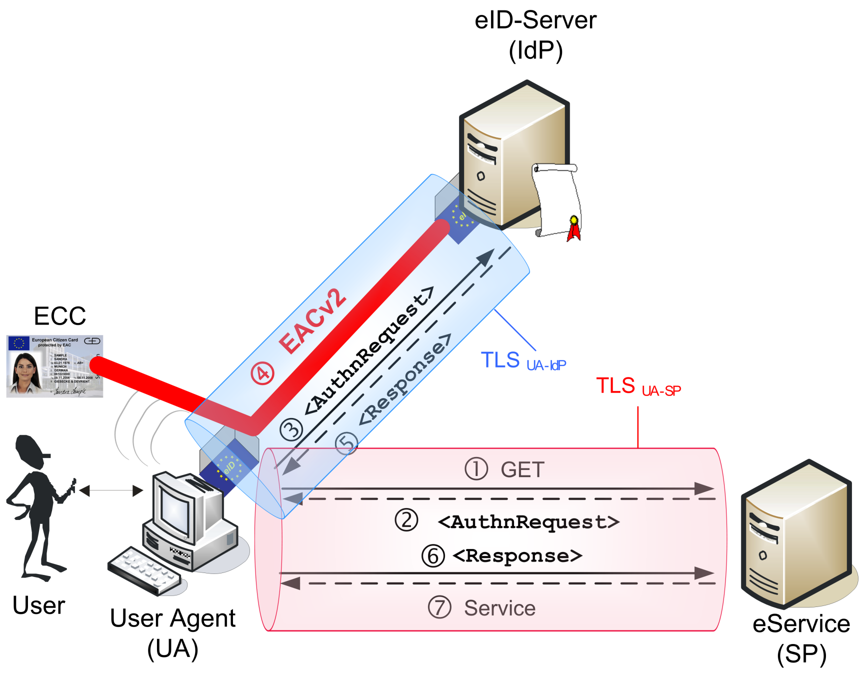


Figure 4: Combined ECC-3 and SAML architecture

The major goal for the integration of the ECC into a SAML-environment is that the Service shall be accessible to the User (Agent) in step (7) if and only if an EAC-based authentication has been successfully performed in step (4). Furthermore it may be desirable to have the option to include cryptographic evidence into the SAML-Assertion transported in steps (5) and (6) such that it can be proved at a later point in time (e.g. at court) that the SAML-Assertion indeed was generated with a valid European Citizen Card (ECC).

However as explained in [SAML-SecP(v2.0)] there are a number of threats against SAML-based solutions, which need to be considered to end up with a secure system. We only consider the man-in-the-middle (MitM) attack here and refer to Section 2.1.3 and [SAML-SecP(v2.0)] for other security aspects related to SAML.

If the TLS-channels are established in an anonymous mode, in which no X.509-certificates are used, it is clear that an attacker may mount a MitM-attack as depicted in Figure 5, steal the SAML-Assertion contain in the `Response`-element in order to impersonate the User at the eService.

In a similar fashion an attacker may mount a MitM-attack, if only the TLS-servers (i.e. the

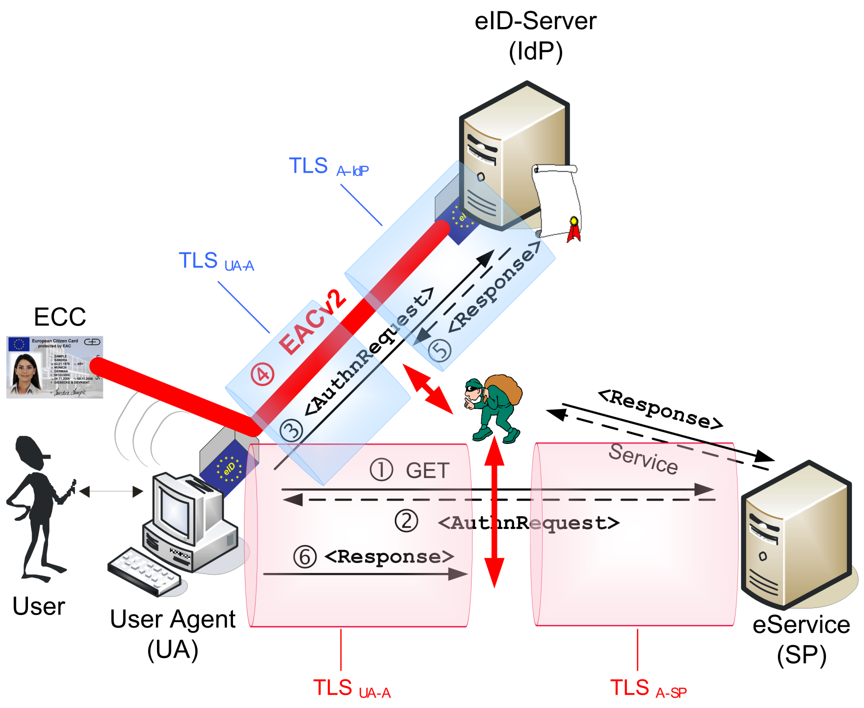


Figure 5: Man-in-the-Middle-Attack against SAML

eID-Server, the eService and the attacker) and are equipped with X.509-certificates and the User is not able to recognize the difference between the certificates presented within the TLS-handshakes. Note that this is a realistic assumption since studies have shown that typical internet users tend to ignore TLS security indicators [DTH06], and that it currently may even be possible to fake trustworthy looking TLS server certificates [SLW09].

3.2 Secure Binding of SAML to TLS

In order thwart attackers, which try to steal a SAML token, e.g. Assertion or an Artifact, one may provide a cryptographic binding between the SAML token and the underlying TLS-channel.

In previous work, we identified three methods to bind SAML tokens to a specific TLS session. By binding the token to the session, the eServer may deduce that the data he sends in response to the SAML token will be protected by the same TLS-channel, and will thus reach the same client who has previously sent the token.

- **TLS Federation [BHS08].** In this approach, the SAML token is sent inside an X.509 client certificate. The SAML token thus may replace other identification data like distinguished names. The certificate has the same validity period as the SAML token.

- **SAML 2.0 Holder-of-Key Web Browser SSO and Assertion Profile [SAML-HoKWebSSO, SAML-HoKAP].** Here again TLS with client authentication is used, but the client certificate does not transport any autorization 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. The security of this approach has independently been analyzed in [GJMS08].

- **Strong Locked Same Origin Policy [GLS2008].** Whereas the previous approaches relied on the server authenticating (in an anonymous fashion) the client, in this approach we strengthen the client 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.

3.3 Secure Binding of TLS to EAC

Since the EAC authentication can be performed over any communication link, it is even possible to successfully complete it over two TLS-channels between the User Agent and the eID-Service with a MitM-attacker in between (cf. Figure 5). Note that the MitM-attack does not affect the EAC-authentication itself, but only allows the attacker to intercept the SAML-Assertion, which is issued as a result of the EAC-authentication. In order to avoid this kind of attack one may include TLS-specific values in the EAC-protocol in order to provide a cryptographic binding between TLS and EAC.

For this purpose we first investigate *which* TLS-specific parameters may be included into the EAC protocol and then we briefly discuss *how* these values may precisely be incorporated into EAC such that the TLS- and EAC-channels are cryptographically tied together.

3.3.1 TLS-specific parameters for potential inclusion in EAC

We consider the following values from a TLS handshake for inclusion in EAC:

- **Certificates or other messages of the TLS-Handshake Protocol.** While it should be easy to access these values using a browser plugin or a server component, it would not be sufficient to use those parameters as they do not depend on both communication partners. Furthermore the used certificates are typically not session specific.

- **Premaster secret.** This value can only be used if a cipher suite using Diffie-Hellman key exchange is chosen. If RSA encryption is used, the MitM-attacker can simply decrypt the premaster secret chosen by the browser, and re-encrypt it for the server.

- **Master secret.** The master secret, or any value derived from it, can be used, since the two nonces sent by browser and server are used to compute it. While a derivation mechanism for the master secret is described in [Resc09] this mechanism does not seem to be supported by popular browsers.

- **Finished message.** Another approach would be to use one of the two Finished messages, since this value is derived from the master secret, and it is sent protected only by the TLS record layer. Thus it should be easy for a browser plugin, or a server component, to access it.
- **Pre-shared key between the eID-Server and eService.** In [BSI-TR-03112-7] it is described how to provide a binding between the two TLS-connections using a pre-shared-key (PSK) known to the eID-Server and the eService. The PSK may be generated by the eID-Server, the eService or both and is transported from the eService to the User Agent over the first TLS-channel (TLS_{UA-SP} in Figure 4) and used for the establishment of the second TLS-channel between the User Agent and the eID-Server (TLS_{UA-IP} in Figure 4) as specified in [RFC4279].

  In addition to the binding of the two TLS-channels the PSK may also be used to provide a binding of the TLS-channels to the EAC-channel.

In particular the last two values seem to fulfill our requirements very well and may serve as input for a binding of TLS to EAC.

3.3.2 Integration of TLS-specific values into EAC

It remains to discuss how the TLS-specific values may be integrated into EAC. For this purpose there are the following general options induced by the structure of the EAC-protocol:

- **Terminal Authentication.** The Terminal Authentication (cf. [BSI-TR-03110(V2.01), Section 4.4]) roughly consists of the following three steps:

  1. As a first step in the Terminal Authentication protocol the ECC verifies the Card-verifiable-Certificate (CVC) of the eID-Server.

     In order to provide a cryptographic link between the X.509 certificate used in TLS and the CVC used in EAC it would be possible to include (a hash value of) one certificate as an extension into the other certificate. For the inclusion of the CVC into an X.509-certificate one may use the `CVCert`-extension defined in [ISO18013-3, Section C.7.2.1]. In order to include the hash value of an X.509-certificate in a CVC it would be necessary to define a corresponding extension in an amendment of [BSI-TR-03110(V2.01), Annex C.3]. On the other side it would – from a theoretical point of view – be possible that the Card-verifiable-Certificates are directly used in TLS in a similar fashion as one may use OpenPGP-keys (cf. [RFC5081]).
  2. Next the eID-Server generates an ephemeral key pair, which is especially used in the Chip Authentication protocol described below. As explained in Section 3.4 the private ephemeral key may be derived from a secret, which is shared by the eService and the eID-Server.

3. Finally a challenge is obtained from the ECC and signed by the eID-Server.

   This challenge contains an identifier derived from the ephemeral PACE-key of the ECC, a nonce generated by the ECC, an identifier derived from the ephemeral public key of the eID-Server generated in the previous step and possibly further "Authenticated Auxiliary Data" (AAD) (cf. [BSI-TR-03110(V2.01), Annex A.6.5]). The AAD are normally used for age verification, document validity verification and community ID verification, but it seems to be possible to use the AAD to convey the TLS-specific value discussed above such that the TLS-channel is cryptographically bound to the EAC-channel, which effectively removes the MitM-attack described above (cf. Figure 5).

- **Chip Authentication.** In the Chip Authentication protocol (cf. [BSI-TR-03110(V2.01), Section 4.3]) the static public key of the ECC and the ephemeral public key of the eID-Server generated in step 2 above is used to agree on a common key, which is used to derive secure messaging keys and authenticate the chip of the ECC. Without significant changing the protocol and the related smart card implementation it seems to be the only option to use the TLS-specific value as seed for the generation of the ephemeral private key of the eID-Server and the keys necessary to verify this construction would provide access to the secure messaging channel between the eID-Server and the ECC. Please refer to Section 3.4 for the use of this feature in the context of SAML.

3.4 Secure Binding of SAML to EAC

For sensitive use cases it may be necessary to enable the eService, which only has access to the SAML-Assertion, to verify that the authentication indeed has been performed using an authentic ECC and that the attributes conveyed in the SAML-Assertion indeed have been read out from the ECC in a secure EAC-session. In order to achieve this a cryptographic binding between SAML and EAC may be constructed as explained in the following.

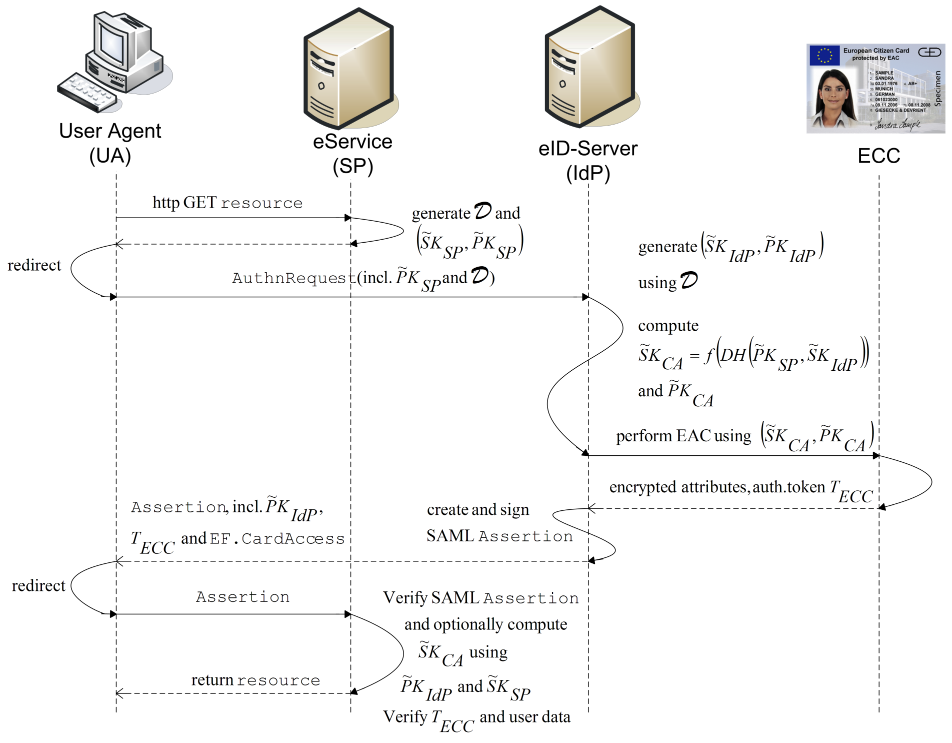
The authentication of the ECC is achieved by the chip authentication protocol, which basically is a Diffie-Hellman (DH) key exchange using static keys on the chip side. The resulting keys are used for secure messaging later on. On the other side the eID-Server would usually generate an ephemeral DH key pair using a random seed. In our case however the ephemeral private key is derived from a shared key which has been agreed upon by the eService and the eID-Server. This allows the eService to add own random data to the key generation process and more importantly it allows the eService to verify that the authentication has been performed with a trustable ECC and that sensitive attributes contained in the SAML-Assertion indeed have been read out from the ECC in a secure EAC-session (see Figure 6).

Before sending the SAML `<AuthnRequest>` to the eID-Server, the eService generates an ephemeral DH key pair $(\tilde{SK}_{SP}, \tilde{PK}_{SP})$ and sends the public key $\tilde{PK}_{SP}$ together with the domain parameters $\mathcal{D}$ within the SAML `<AuthnRequest>` to the eID-Server. The additional data may be placed within the `<AuthnRequest>.<RequestedAuthnCon-`

`text>.<AuthnMethod>.<AsymmetricKeyAgreement>` structure for example.

Upon receiving the `<AuthnRequest>` the eID-Server also generates an ephemeral DH key pair $(\tilde{SK}_{IdP}, \tilde{PK}_{IdP})$ using the domain parameters $\mathcal{D}$ chosen by the eService. Using $\tilde{PK}_{SP}$ and $\tilde{SK}_{IdP}$, the eID-Server calculates the common key which is used to derive the ephemeral private key $\tilde{SK}_{CA}$ and the corresponding $\tilde{PK}_{CA}$, which is used in the Chip Authentication protocol.

After the eID-Server has successfully performed the EAC protocol with the ECC, he received the data $(\mathcal{D}, PK_{ECC}, EF.CardSecurity, r_{ECC}, T_{ECC})$ from the ECC, which can be used to verify the genuineness of the ECC.



User Agent (UA)
eService (SP)
eID-Server (IdP)
ECC
http GET resource
generate $\mathcal{D}$ and $(\tilde{SK}_{SP}, \tilde{PK}_{SP})$
redirect
AuthnRequest (incl. $\tilde{PK}_{SP}$ and $\mathcal{D}$)
generate $(\tilde{SK}_{IdP}, \tilde{PK}_{IdP})$ using $\mathcal{D}$
compute $\tilde{SK}_{CA} = f(DH(\tilde{PK}_{SP}, \tilde{SK}_{IdP}))$ and $\tilde{PK}_{CA}$
perform EAC using $(\tilde{SK}_{CA}, \tilde{PK}_{CA})$
encrypted attributes, auth.token T_{ECC}
Assertion, incl. $\tilde{PK}_{IdP}$, T_{ECC} and EF.CardAccess
create and sign SAML Assertion
redirect
Assertion
Verify SAML Assertion and optionally compute $\tilde{SK}_{CA}$ using $\tilde{PK}_{IdP}$ and $\tilde{SK}_{SP}$ Verify T_{ECC} and user data
return resource

Figure 6: Command Flow for eService Authentication Token verification

Within the SAML element `<Response>.<Assertion>.<AuthnStatement>.<AuthnContext>` – which is an instance of [SAML-Auth(v2.0), Section 3.4.15] – the necessary verification data can be placed and hence be made available to the eService. Using $\tilde{PK}_{IdP}$ and $\tilde{SK}_{SP}$ the eService is able to compute the private key $\tilde{SK}_{CA}$ used in the Chip Authentication protocol. Afterwards it may use $\tilde{SK}_{CA}$ and PK_{ECC} to compute the secure messaging keys and hence verify the validity of the authentication token T_{ECC}.

Since the eService now has the secure messaging keys of the EAC-channel, it would be possible that the eID-Server does not decrypt the data received from the ECC, but instead places the secure messaging cryptograms received from the ECC within an `EncryptedAttribute`-element within the Assertion. To retrieve the plain value of the attributes, the eService needs to decrypt the `EncryptedAttribute`-element with the derived secure messaging key.

3.5 Discussion of different approaches and recommendations

In this section it remains to discuss the different approaches presented above and derive recommendations for the secure integration of the European Citizen Card into a SAML-environment.

As the mechanisms mentioned in Section 3.2, such as [SAML-HoKWebSSO, SAML-HoKAP] for example, are independent from the applied authentication protocol they may of course be used in the ECC-context.

In case of an ECC which supports the EAC protocol however it is possible to provide a tighter and probably more secure binding between EAC, TLS and SAML.

Among the different options discussed in Section 3.3 one may in particular include TLS-specific values as additional ”Authenticated Auxiliary Data” (AAD) into the Terminal Authentication step within the EAC protocol in order to provide a strong binding between TLS and EAC. If there is already a pre-shared key (PSK) between the eID-Server, the eService and the User Agent as required by [BSI-TR-03112-7] one may include (the hash value of) this value as AAD in EAC. Alternatively one may use the (hash value of the concatenation of the) Finished Messages of the TLS-channels as input to the EAC-protocol. While the eID-Server has direct access to the Finished Messages of TLS_{UA-IdP} the corresponding value for TLS_{UA-IdP} would need to be transported in encrypted form to the eID-Server and may be included in the optional `<Extensions>`-element within `<AuthnRequest>`.

Whether it makes sense to introduce a cryptographic link between the CVC used for EAC and the X.509-certificates used for TLS mainly depends on organizational aspects such as the respective certificate lifetime and involved enrollment procedures.

In order to provide a direct binding between SAML and EAC and especially if the eService requires a proof that the authentication was performed with a trustable ECC, it is highly recommendable to use the mechanism introduced in Section 3.4. This proposal seems to be especially attractive from a practical point of view as it may help to solve liability issues introduced by the delegation of the sensitive authentication step to the eID-Server.

Finally for maximum security one may combine the different proposals and link SAML to TLS (cf. Section 3.2), TLS to EAC (cf. Section 3.3) and SAML to EAC (cf. Section 3.4).

4 Conclusion

Based on the discussion in the previous sections it seems that the integration of the European Citizen Card into a SAML-environment has the potential to solve many open issues related to the acceptance of ECC based authentication protocols, fast deployment and easy integration into existing web service infrastructures, which already (are about to) use SAML.

However, the slightly increased complexity of the system introduces additional threats as an attacker may for example act as Man-in-the-Middle and steal the SAML-Assertion and finally impersonate the User which has been securely authenticated based on the ECC. In order to prevent such attacks various mechanisms have been proposed which provide a cryptographic binding between SAML, TLS and EAC. Furthermore the binding between SAML and EAC may be helpful to solve liability issues due to the introduction of the eID-Server acting as trusted third party.

To sum up we solved security problems which are also present in many other Federated Identity Management scenarios, we greatly simplify the introduction of ECC into existing web service infrastructures, and we introduced an approach which may help to solve liability issues related to the delegation of the sensitive authentication step.

References

[2006/123/EC] *Directive 2006/123/EC of the European Parliament and the Council of 12 December 2006 on Services in the Internal Market.* Official Journal of the European Union, L 376/36, 27.12.2006. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2006:376:0036:0068:EN:PDF, 2006.

[ACC+08] 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. http://www.ai-lab.it/armando/pub/fmse9-armando.pdf, 2008.

[BFK09] Jens Bender, Marc Fischlin, and Dennis Kügler. *Security Analysis of the PACE Key-Agreement Protocol.* to appear at 12th International Information Security Conference (ISC 2009), September 2009.

[BHS08] 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.

[BSI-TR-03110(V1.11)] Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik). *Advanced Security Mechanism for Machine Readable Travel Documents - Extended*

Access Control (EAC). Technical Directive (BSI-TR-03110), Version 1.11. http://www.bsi.bund.de/literat/tr/tr03110/TR-03110_v111.pdf, 2008.

[BSI-TR-03110(V2.01)] 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 2.01. http://www.bsi.bund.de/english/publications/techguidelines/tr03110/TR-03110_v201.pdf, 2009.

[BSI-TR-03112-7] FEDERAL OFFICE FOR INFORMATION SECURITY (BUNDESAMT FÜR SICHERHEIT IN DER INFORMATIONSTECHNIK). *eCard-API-Framework – Protocols*. Technical Directive (BSI-TR-03112), Version 1.1, Part 7. http://www.bsi.bund.de/literat/tr/tr03112/, 2009.

[CEN14890-1] COMITÉ EUROPÉEN DE NORMALISATION (CEN). *Application Interface for smart cards used as Secure Signature Creation Devices - Part 1: Basic services*. Preliminary European Norm, 2008.

[CEN14890-2] COMITÉ EUROPÉEN DE NORMALISATION (CEN). *Application Interface for smart cards used as Secure Signature Creation Devices - Part 2: Additional Services*. Preliminary European Norm, 2008.

[CEN15480-1] COMITÉ EUROPÉEN DE NORMALISATION (CEN). *Identification card systems - European Citizen Card - Part 1: Physical, electrical and transport protocol characteristics*. CEN/TS 15480-1 (Technical Specification), 2007.

[CEN15480-2] COMITÉ EUROPÉEN DE NORMALISATION (CEN). *Identification card systems - European Citizen Card - Part 2: Logical data structures and card services*. CEN/TS 15480-2 (Technical Specification), 2007.

[CEN15480-3] COMITÉ EUROPÉEN DE NORMALISATION (CEN). *Identification card systems - European Citizen Card - Part 3: European Citizen Card Interoperability using an application interface*. CEN 15480-3 (Working Draft), 2008.

[CEN15480-4] COMITÉ EUROPÉEN DE NORMALISATION (CEN). *Identification card systems - European Citizen Card - Part 4: Recommendations for European Citizen Card issuance, operation and use*. CEN 15480-4 (Working Draft), 2008.

[DTH06] RACHNA DHAMIJA, J. D. TYGAR, and MARTI HEARST. *Why phishing works*. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 581–590 (ACM, 2006). http://graphics8.nytimes.com/images/blogs/freakonomics/pdf/Why_Phishing_Works-1.pdf.

[GJMS08] 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, 2008).

[Gaje08] 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 (2008).

[GLS2008] 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).

[GPS05] 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, pages 489–508 (Berlin, Germany, 2005). http://eprint.iacr.org/2005/127.pdf.

[Gros03] 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.

[GrPf06] THOMAS GROSS and BIRGIT PFITZMANN. *SAML Artifact Information Flow Revisited*. In *In IEEE Workshop on Web Services Security (WSSS)*, pages 84–100 (2006).

[ISO18013-3] *ISO/IEC 18013-1: Personal Identification – ISO Compliant Driving Licence – Part 3: Access control, authentication and integrity validation*. International Standard, 2009.

[ISO24727-1] *ISO/IEC 24727-1: Identification cards – Integrated circuit cards programming interfaces – Part 1: Architecture*. International Standard, 2007.

[ISO24727-2] *ISO/IEC 24727-2: Identification cards – Integrated circuit cards programming interfaces – Part 2: Generic Card Interface*. International Standard, 2008.

[ISO24727-3] *ISO/IEC 24727-3: Identification cards – Integrated circuit cards programming interfaces – Part 3: Application programming interface*. International Standard, 2008.

[ISO24727-4] *ISO/IEC 24727-4: Identification cards – Integrated circuit cards programming interfaces – Part 4: API Administration*. International Standard, 2008.

[ISO7816-8] *ISO/IEC 7816-8: Identification cards – Integrated circuit cards – Part 8: Commands for security operations*. International Standard, 2004.

[NIST-800-63] 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.

[PAOS(v1.0)] ROBERT AARTS. *Liberty Reverse HTTP Binding for SOAP Specification*. Liberty Alliance Specification, Version 1.0. https://www.projectliberty.org/liberty/content/download/2008/13941/file/liberty-paos-v1.0.pdf, 2003.

[PfWa03] BIRGIT PFITZMANN and MICHAEL WAIDNER. *Analysis of liberty single-sign-on with enabled clients*. *Internet Computing, IEEE*, volume 7(6):38–44. http://www.cs.ru.nl/~jhh/pub/secsem/pfitzmann2003liberty-single-sign-on.pdf, 2003.

[Resc09] E. RESCORLA. *Keying Material Exporters for Transport Layer Security (TLS)*. IETF Internet Draft, v6. http://www.ietf.org/id/draft-ietf-tls-extractor-06.txt, July 2009.

[RFC2616] 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. http://www.ietf.org/rfc/rfc2616.txt, Juni 19969.

[RFC2903] C. DE LAAT, G. GROSS, L. GOMMANS, J. VOLLBRECHT, and D. SPENCE. *Generic AAA Architecture*. Request For Comments – RFC 2903. http://www.ietf.org/rfc/rfc2903.txt, August 2000.

[RFC3369] R. HOUSLEY. *Cryptographic Message Syntax (CMS)*. Request For Comments – RFC 3369. http://www.ietf.org/rfc/rfc3369.txt, August 2002.

[RFC4279] P. ERONEN and H. TSCHOFENIG. *Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)*. Request For Comments – RFC 4279. http://www.ietf.org/rfc/rfc4279.txt, December 2005.

[RFC5081] N. MAVROGIANNOPOULOS. *Using OpenPGP Keys For Transport Layer Security Authentication*. Request For Comments – RFC 5081. http://www.ietf.org/rfc/rfc5081.txt, November 2007.

[RFC5246] T. DIERKS and E. RESCORLA. *The Transport Layer Security (TLS) Protocol Version 1.2*. Request For Comments – RFC 5246. http://www.ietf.org/rfc/rfc5246.txt, August 2008.

[SAML-Auth(v2.0)] 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. http://docs.oasis-open.org/security/saml/v2.0/saml-authn-context-2.0-os.pdf, 2005.

[SAML-Bind(v2.0)] 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. http://docs.oasisopen.org/security/saml/v2.0/saml-bindings-2.0-os.pdf, 2005.

[SAML-Conf(v2.0)] PRATEEK MISHRA, ROB PHILPOTT, and EVE MALER. *Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS Standard, 15.03.2005. http://docs.oasis-open.org/security/saml/v2.0/saml-conformance-2.0-os.pdf, 2005.

[SAML-Glos(v2.0)] JEFF HODGES, ROB PHILPOTT, and EVE MALER. *Glossary for the OASIS Security Assertion Markup Language (SAML) V2.0*. OASIS Standard, 15.03.2005. http://docs.oasis-open.org/security/saml/v2.0/saml-glossary-2.0-os.pdf, 2005.

[SAML-HoKAP] TOM SCAVO. *SAML V2.0 Holder-of-Key Assertion Profile*. OASIS Committee Draft 02, 05.07.2009. http://www.oasis-open.org/committees/download.php/33236/sstc-saml2-holder-of-key-cd-02.pdf, 2009.

[SAML-HoKWebSSO] N. KLINGENSTEIN. *SAML V2.0 Holder-of-Key Web Browser SSO Profile*. OASIS Committee Draft 02, 05.07.2009. http://www.oasis-open.org/committees/download.php/33239/sstc-saml-holder-of-key-browser-sso-cd-02.pdf, 2009.

[SAML-Meta(v2.0)] 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. http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf, 2005.

[SAML-Prof(v2.0)] 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. http://docs.oasis-open.org/security/saml/v2.0/saml-profiles-2.0-os.pdf, 2005.

[SAML-Resp] 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. http://www.oasis-open.org/committees/download.php/11191/sstc-gross-sec-analysis-response-01.pdf, 2005.

[SAML-SecP(v2.0)] 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. http://docs.oasis-open.org/security/saml/v2.0/saml-sec-consider-2.0-os.pdf, 2005.

[SAML(v1.0)] PHILLIP HALLAM-BAKER and EVE MALER. *Assertions and Protocol for the OASIS Security Assertion Markup Language (SAML)*. OASIS Standard, 05.11.2002. http://www.oasis-open.org/committees/download.php/2290/oasis-sstc-saml-1.0.zip, 2002.

[SAML(v1.1)] 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. http://www.oasis-open.org/committees/download.php/3406/oasis-sstc-saml-core-1.1.pdf, 2003.

[SAML(v2.0)] 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. http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf, 2005.

[SLW09] Marc Stevens, Arjen Lenstra, and Benne de Weger. *Chosen-prefix Collisions for MD5 and Applications*. Submitted to Journal of Cryptology. https://documents.epfl.ch/users/l/le/lenstra/public/papers/lat.pdf, June 2009.

[SOAP(v1.1)] 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. http://www.w3.org/TR/2000/NOTE-SOAP-20000508, May 2000.

[X.509:05] ITU-T. *ITU-T Recommendation X.509 (2005) - ISO-IEC 9594-8:2005*. Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks. http://www.itu.int/rec/T-REC-X.509-200508-I/en, August 2005.

[XML-DSig] D. Eastlake, J. Reagle, and D. Solo. *XML-Signature Syntax and Processing*. W3C Recommendation. http://www.w3.org/TR/xmldsig-core/, June 2008.