New Authentication concepts for electronic Identity Tokens

Jan Eichholz[1] · Dr. Detlef Hühnlein[2] · Dr. Gisela Meister[1] · Johannes Schmölz[3]

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

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

[3] Hochschule Coburg
Friedrich-Streib-Straße 2, 96450 Coburg, Germany
schmoelz@hs-coburg.de

Abstract

The national funded project **[BioP@ss]** researches the possibilities of an IP based smart card interface based on the international smart card application interface standards **[CEN 15480]** and **[ISO/IEC 24727]**. Instead of the classical APDU based communication a TCP/IP based web service communication with the smart card is established. This solution offers the benefit that this interface relies on well established standardized Internet protocols and hence reduces the necessity of an intermediate middleware implementation which translates web service calls into APDU's. Additionally, we define a **[SAML(v2.0)]** profile, which allows the implementation of an Identity Provider directly on a smart card.

1 Introduction

In the area of electronic identity management there are currently two major European projects. While the EU funded project **[STORK]** is dealing with the interoperability aspects of the existing eID solutions in Europe, the national funded project **[BioP@ss]** researches new concepts for smart cards to enhance the performance and interoperability for the next generation of electronic identity tokens and electronic passports. The German part of the **[BioP@ss]** project is funded by the German Federal Ministry of Education and Research.

Within the **[BioP@ss]** project Giesecke & Devrient researches the possibilities of a new smart card interface based on a HTTP(S) communication using SOAP based web services as interface to the outside world.

Federated Identity Management solutions which are often based on the Security Assertion Markup Language (SAML) **[SAML(v2.0)]**, are increasingly used in practice as they allow to implement Single Sign-On, facilitate the integration of individual Service Providers and heterogeneous national Identity Management solutions using so called Pan-European Proxy Ser-

vices (PEPS) (e.g. **[STORK]**). Today's IDM solutions integrating with an eID offer the main drawback of introducing an Identity Provider entity, which may break the end-to-end security. Therefore we sketch a solution, how a SAML Identity Provider can be realized directly on a smart card.

The rest of the paper is structured as follows: Section 2 contains the necessary background and motivation. Within Section 3 we describe the Service Access Layer as a new smart card interface and introduce new concepts for authentication protocols based on web services. Section 4.4 proposes a new eID-specific SAML-profile. Finally, in Section 5 we draw conclusions.

2 Background and motivation

This section carries together the background information, which is necessary to understand the new concepts presented in the sequel.

2.1 Standardized interfaces in the context of electronic Identity Cards

During the last years, the smart card middleware standards **[ISO/IEC 24727]** and **[CEN 15480]** (European Citizen Card) have been developed in order to allow a Client Application to access arbitrary cryptographic tokens through a generic Service Access Interface defined in Part 3 of **[ISO/IEC 24727]**.

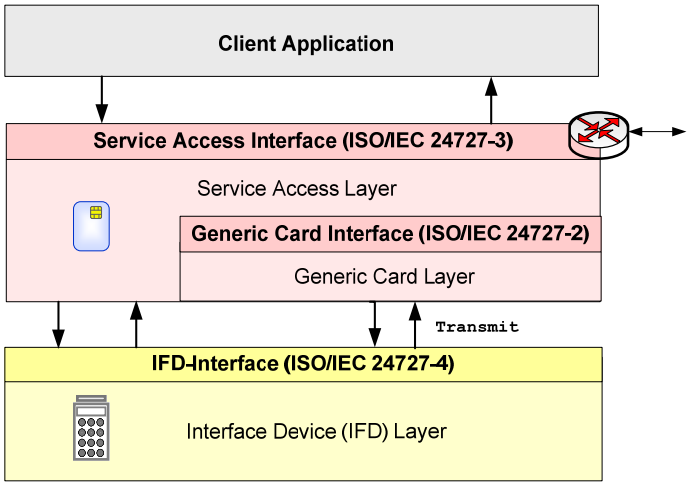

Figure 1: ISO/IEC 24727 Architecture

This interface comprises generic functions which allow to establish (cryptographically protected) connections to card-applications, manage card-applications, store and retrieve data, perform cryptographic operations, manage the related key material (so called Differential-Identities (DID)) and manage access rights for data, keys and services provided by card-applications.

The Service Access Layer (SAL) maps the generic requests at the Service Access Interface to APDUs of the Generic Card Interface defined in Part 2 of **[ISO/IEC 24727]**, which allows a subset of the commands and options defined in **[ISO/IEC 7816]** (Part 4, 8 and 9). If the cryptographic token does not support those standard-commands directly they may be translated by

the Generic Card Layer before they are sent to the Interface Device (IFD) Layer using the <Transmit>-command defined in Part 4 of **[ISO/IEC 24727]**.

An important feature of the **[ISO/IEC 24727]** architecture is that it supports arbitrary authentication protocols using the generic <DIDAuthenticate>-function, which contains an "open type", which allows to "plug in" protocol-specific elements and hence support arbitrary authentication protocols.

The **[ISO/IEC 24727]** standard defines different possibilities how the components can be distributed among the network. In this paper we concentrate on the so called ICC-resident-stack configuration, which assumes that the SAL is implemented directly on the smart card. As a consequence, only a minimal IFD-layer and no GCAL at all is necessary.

The European Citizen Card standard **[CEN 15480]** extends **[ISO/IEC 24727]** with respect to specific requirements in the context of electronic health and identity cards. Furthermore it clarifies the communication processes in a distributed middleware scenario, where a server and a client PC together with an eID are involved.

2.2 Java Card 3.0 connected

The Java Card 3.0 (**[JC30]**) connected platform provides smart cards with improved connectivity. Within the BioP@ss project the most interesting part of JC3.0 is the Servlet-API, which is very similar to the Servlet-API defined within the Java Enterprise Edition.

With the help of the Servlet-API it's possible to implement a Servlet running on the card, which can react on HTTP commands. This serves as the basis to add a Service Access Layer Web Service to the smart card.

2.3 Existing and emerging SAML-related profiles

Currently there are the following SAML-related profiles, which need to be considered here:

- ***Web Browser SSO Profile*** **[SAML-Prof(v2.0)]** (Section 4.1)

  can be used with arbitrary web browsers serving as User Agents, as the relevant SAML-messages (AuthnRequest and Response) are transported using plain HTTP and the following bindings: *HTTP Redirect Binding* **[SAML-Bind(v2.0)]** (Section 3.4), *HTTP POST Binding* **[SAML-Bind(v2.0)]** (Section 3.5) or *HTTP Artifact Binding* **[SAML-Bind(v2.0)]** (Section 3.6).

- ***Enhanced Client or Proxy (ECP) Profile*** **[SAML-Prof(v2.0)]** (Section 4.2)

  requires that the User Agent supports Web Service interfaces in which the SAML-messages are received using the *Reverse SOAP (PAOS) Binding* **[SAML-Bind(v2.0)]** (Section 3.3) and send using *SOAP Binding* **[SAML-Bind(v2.0)]** (Section 3.2).

- ***Holder-of-Key Web Browser SSO Profile*** **[SAML-HoK]**

  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.

- ***PEPS-interface specification developed in STORK-project*** **[STORK-D.5.8.1b]**

defines a SAML-based interface between the Pan-European Proxy Service (PEPS) in the country of the Service Provider (S-PEPS) and the corresponding PEPS located in the citizen country (C-PEPS). Section 7 of **[STORK-D.5.8.1b]** specifies a set of STORK-specific attributes, which seem to be unrelated to other existing attribute profiles.

3 The Service Access Layer as interoperable smart card interface

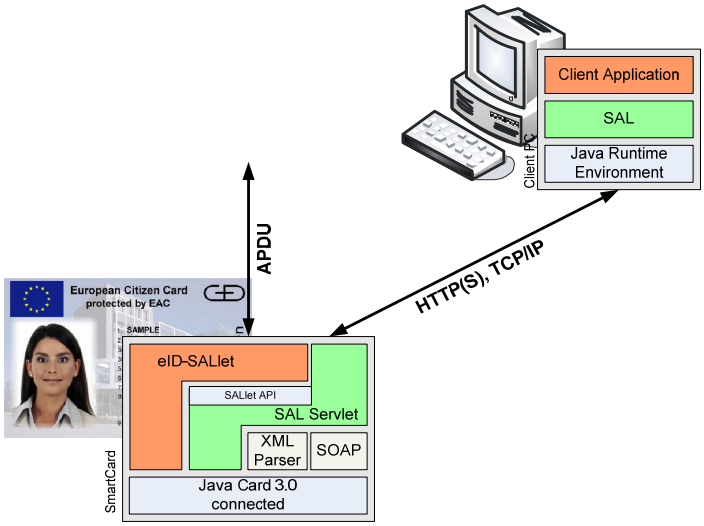

Figure 2 - The SAL-on-card architecture

The SAL implementation can be realized on top of a Java Card 3.0 connected operating system (see Figure 2). Hence, the SAL can be seen as a new interface to the smart card using well known Internet protocols like TCP/IP and HTTP(S) for transport purposes. This approach highly increases the interoperability of the complete system. The communication is based on HTTP(S) and the smart card offers a Servlet (SALServlet), which reacts on web service requests from the outside world. The web service binding of the Service Access Layer is implemented according to **[CEN 15480]** part 3.

Additionally, the SALlet concept (SALlet-API) allows an easy implementation and integration of new applets onto this platform.

4 New Authentication Concepts

ISO/IEC 24727-3 defines a set of generic authentication mechanisms. All these protocols assume the “classical” infrastructure, meaning that the smart card interface is APDU based and the middleware between the application and the smart card is responsible for translating high level Service Access Layer calls into APDU’s. As a consequence, all these protocols assume after a successful protocol execution a channel protection by the means of secure messaging, which is only valid in terms of APDU’s.

[CEN 15480] already contains a web service binding for some protocols. Additional web service bindings are available within **[TR-03112(v1.1)]**. In the following, we will investigate the web service binding of the Extended Access Control (EAC) protocol in an IP based smart card scenario.

4.1 EAC Web Service Binding

The web service binding defined in **[CEN 15480]** assumes a stack model, which is a combination of the Remote-loyal and Remote-ICC-stack.

In the ICC-resident stack scenario, the defined authentication data structures are not applicable. In the non-ICC-resident-stack scenario, the authentication protocol flow (PACE, Terminal Authentication and Chip Authentication) is coded in terms of APDU's and the APDU communication is encapsulated by the Service Access Layer.

In the case of the ICC-resident-stack the authentication protocol flow has to be encoded in XML-structures as input for a `DIDAuthenticate` command.

The necessary structures for the different `DIDAutenticate` calls are described in Figure 3, which are based – if possible – on the definitions provided in **[TR-03112(v1.1)]**. The protocol flow is according to the definitions in **[TR-03110(V2.02)]**.

```
<complexType name="EAC_oncard_InputType_1">
	<complexContent>
		<restriction base="iso:DIDAuthenticationDataType">
		</restriction>
	</complexContent>
</complexType>

<complexType name="EAC_oncard_OutputType_1">
	<complexContent>
		<restriction base="iso:DIDAuthenticationDataType">
			<sequence>
				<element name="EncryptedNonce" type="hexBinary"
					maxOccurs="1" minOccurs="1" />
			</sequence>
		</restriction>
	</complexContent>
</complexType>

<complexType name="EAC_oncard_InputType_2">
	<complexContent>
		<restriction base="iso:DIDAuthenticationDataType">
			<sequence>
				<element name="DHPublicKey" type="hexBinary"
					maxOccurs="1" minOccurs="1" />
			</sequence>
		</restriction>
	</complexContent>
</complexType>

<complexType name="EAC_oncard_OutputType_2">
	<complexContent>
		<restriction base="iso:DIDAuthenticationDataType">
			<sequence>
				<element name="DHPublicKey" type="hexBinary"
					maxOccurs="1" minOccurs="1" />
			</sequence>
		</restriction>
	</complexContent>
</complexType>

<complexType name="EAC_oncard_InputType_3">
	<complexContent>
		<restriction base="iso:DIDAuthenticationDataType">
			<sequence>
				<element name="AuthToken" type="hexBinary"
					maxOccurs="1" minOccurs="0" />
			</sequence>
		</restriction>
	</complexContent>
</complexType>
```

```
      <complexType name="EAC_oncard_OutputType_3">
            <complexContent>
                  <restriction base="iso:DIDAuthenticationDataType">
                        <sequence>
                              <element name="AuthToken" type="hexBinary"
                                    maxOccurs="1" minOccurs="0" />
                              <element name="RetryCounter"
                                    type="nonNegativeInteger" maxOccurs="1" mi-
nOccurs="0" />
                              <element name="EFCardAccess" type="hexBinary"
                                    maxOccurs="1" minOccurs="0" />
                              <element name="IDPICC" type="hexBinary"
                                    maxOccurs="1" minOccurs="0" />
                              <element name="Challenge" type="hexBinary"
                                    maxOccurs="1" minOccurs="0" />
                        </sequence>
                  </restriction>
            </complexContent>
      </complexType>

      <complexType name="EAC_oncard_InputType_4">
            <complexContent>
                  <restriction base="iso:DIDAuthenticationDataType">
                        <sequence>
                              <sequence maxOccurs="unbounded" minOccurs="0">
                                    <element name="Certificate" type="hexBinary"
                                          maxOccurs="1" minOccurs="1" />
                              </sequence>
                              <element name="EphemeralPublicKey"
type="hexBinary"
                                    maxOccurs="1" minOccurs="1" />

                              <element name="Signature" type="hexBinary"
                                    maxOccurs="1" minOccurs="1" />
                        </sequence>
                  </restriction>
            </complexContent>
      </complexType>

      <complexType name="EAC_oncard_OutputType_4">
            <complexContent>
                  <restriction base="iso:DIDAuthenticationDataType">
                        <sequence>
                              <element name="EFCardSecurity" type="hexBinary" />
                              <element name="AuthenticationToken"
                                    type="hexBinary" />
                              <element name="Nonce" type="hexBinary" />
                        </sequence>
                  </restriction>
            </complexContent>
      </complexType>

      <complexType name="EAC_oncard_AdditionalInputType">
            <complexContent>
                  <restriction base="iso:DIDAuthenticationDataType">
                        <sequence>
                              <element name="Signature" type="hexBinary"
                                    maxOccurs="1" minOccurs="1" />

                        </sequence>
                  </restriction>
            </complexContent>
      </complexType>
```

Figure 3 - Authentication Protocol Data Structures

4.2 Path Protection based on XML and WS Secure Conversation

In the case of the ICC-resident stack, the smart card interface is based on Web Services, which uses standard TCP/IP and HTTP as bearer. Hence the secure messaging mechanism has to be replaced by an analogous mechanism for web services. One possible solution is the usage of **[XMLEnc]** and **[XMLSig]** as in **[WS-SecCon(v1.4)]** for the protection of the communication channel. The session key generated within the authentication protocol execution of the EAC protocol can be used to derive the session keys used afterwards.

4.3 Path protection based on an EAC-TLS cipher suite

An alternative approach to the solution proposed in chapter 4.2 is the definition of a new TLS cipher suite, which re-uses the privacy concepts of the Extended Access Control protocol (EAC). This offers the benefit, that no additional binding of the different communication channels (TLS and SOAP) is necessary. The drawback is, that the existing XML-based SAL-API (e.g. `DIDAuthenticate`) is not applicable.

4.4 Integrating eID and SAML

Within this chapter we briefly discuss options for the integration of eID and SAML.

4.4.1 Naïve integration using Web Browser SSO Profile

An obvious integration approach is to use the SAML Web Browser SSO Profile **[SAML-Prof(v2.0)]** (Section 4.1) and only use the eID to perform the authentication. As the eID-specific part of the User Agent is activated by the Identity Provider in this case, the message flow is far from being optimal and especially susceptible to Man-in-the-Middle-attacks (cf. **[EHS09]**).

4.4.2 An ECP-based SAML-profile for eID integration

Because the eID-enabled User Agent already supports Web Service interfaces with SOAP and PAOS binding, it is more natural to use the Enhanced Client or Proxy (ECP) Profile defined in **[SAML-Prof(v2.0)]** (Section 4.2) and let the Service Provider activate the User Agent to avoid unnecessary and security critical redirects. For this purpose we define a few additional eID-specific elements, which may appear in the `<samlp:Extension>` element of `<AuthnRequest>` (cf. Section 4.4.2.1).

In order to avoid Man-in-the-Middle-attacks the proposed profile *may* be combined with the **[SAML-HoK]**-profile such that Man-in-the-Middle-attacks can even be avoided if there is no eID-specific CV-PKI, which allows the User Agent to recognize trustworthy TLS-certificates. If Man-in-the-Middle-attacks on TLS are already prevented by authentication protocol specific means, there is no need for using the mechanisms defined in **[SAML-HoK]**.

Finally we propose to use the `<samlp:NameIdPolicy>`-element to indicate that a sector-specific identifier (cf. **[STORK-D.5.8.1b]**, Table 25 and **[LHP02]**, Section 3.2) is to be produced by combining some token specific "Source ID" with the `SPNameQualifier`-attribute provided by the Service Provider.

4.4.2.1 Authentication Request

As discussed above, it is beneficial to extend the <AuthnRequest> structure to minimize the communication overhead. In the following sub-sections we define extension elements, which may be placed within the <samlp:Extension> tag to allow an efficient SAML authentication process based on an eID token.

4.4.2.2 Requested Attributes

The following structure is somewhat similar to the <AttributeQuery>-element defined in **[SAML(v2.0)]** (Section 3.3.2.3) and simply contains a sequence of one or more <Attribute>-elements as defined in **[SAML(v2.0)]** (Section 2.7.3.1). Furthermore we propose to use an additional optional attribute eid:Required of type="boolean", which may be present with a value of true in order to indicate that the requested attribute is required and that the User would need to cancel the entire authentication and identification process if it does not want to disclose this attribute.

```
<element name="RequestedAttributes" type="eid:RequestedAttributesType" />

<complexType name="RequestedAttributesType">
      <sequence>
            <element ref="saml:Attribute" minOccurs="1"
                  maxOccurs="unbounded"/>
      </sequence>
</complexType>
```

<saml:Attribute>

This element is defined in **[SAML(v2.0)]** (Section 2.7.3.1) and specifies the requested attribute.

4.4.2.3 Authentication Protocol Data

The authentication protocol data structure may be used to include authentication protocol specific data in an <AuthnRequest>-element.

This element may contain arbitrary authentication protocol specific data and is specified as follows:

```
<element name="AuthenticationProtocolData"
      type="iso:DIDAuthenticationDataType" />
```

The type DIDAuthenticationDataType is defined in Part 3 of **[CEN 15480]** as abstract data type, which may contain arbitrary authentication protocol specific elements. This type is used as generic template for the specification of protocol specific data types such that it is possible to use DIDAuthenticate with arbitrary authentication protocols.

Once the User Agent receives an <AuthnRequest> containing an AuthenticationProtocolData-element it tries to find a matching token, which contains a DID with the required protocol and executes the protocol using the provided AuthenticationProtocolData. The result of this authentication step is then placed in the same element of the <AuthnRequest> structure, which is forwarded to the Identity Provider, which may reside on the smart card (cf. Section 4.4.3).

If additional protocol steps are necessary to complete the authentication protocol, the SOAP channel between the User Agent and the Identity Provider can be used to convey additional `DIDAuthenticate` commands directly. Note that in this case the User Agent must include appropriate Differential Identity Information in the `<samlp:Extension>` element as explained in Section 4.4.2.4.

Finally consider an example in which the Service Provider specifies that the EAC-protocol **[TR-03110(V2.02)]** must be used for authentication. Then the `AuthnRequest` from the Service Provider to the User Agent would contain an `AuthenticationProtocolData`-element of type `EAC1InputType` (see **[TR-03112(v1.1)]**, which would trigger the User Agent to obtain the user consent and perform the PACE-protocol with an appropriate eID. Furthermore the `AuthnRequest` message from the User Agent to the Identity Provider, which may reside on the smart card, would contain appropriate DID-information (cf. Section 4.4.2.4) and an `AuthenticationProtocolData`-element of type `EAC1_oncard_InputType` (see Figure 3), which triggers the execution of the local PACE protocol. Subsequent `DIDAuthenticate` requests are sent to the card to perform the remaining steps of the EAC protocol. Finally the requested attributes can be retrieved from the eID-token and an `Assertion` is produced, which is returned in the `Response`-element.

4.4.2.4 Differential Identity Information

If the Identity Provider needs to perform a complex authentication protocol and send additional `DIDAuthenticate` commands to the User Agent it requires corresponding Differential Identity Information, which may consist of the following elements, which are defined using types standardized in **[CEN 15480]**:

```
<element name="ConnectionHandle" type="iso:ConnectionHandleType" />
<element name="DIDName" type="iso:NameType" />
<element name="DIDScope" type="iso:DIDScopeType" />
```

`<ConnectionHandle>`

This element is of type `iso:ConnectionHandleType`, which is defined in **[CEN 15480]** and contains a handle for the connection to a card application. If the User Agent is able to recognize the type of a connected token and – possibly with assistance by the User – select an appropriate eID-token, which allows to perform the requested authentication protocol or fulfil the requested Identity Assurance Level, there may be a `ConnectionHandle`-element which should contain the `RecognitionInfo` child element when it is sent to an off-card Identity Provider.

`<DIDName>`

This element is of type `iso:NameType`, which is defined in **[CEN 15480]** and contains the name of a Differential Identity (DID) within the application addressed by the `<ConnectionHandle>`, which is to be used for performing (the remaining part of) a specific authentication protocol selected by the Service Provider or the User Agent.

`<DIDScope>`

This element is of type `iso:DIDScopeType`, which is defined in **[CEN 15480]** and may be used to remove ambiguities, if there are two DIDs (a local and a global) with the same name.

The `DIDName` and `DIDScope` element may only appear at most once in an `AuthnRequest`.

4.4.2.5 Response

In case of success the `Response`-element will contain an `Assertion`, which in turn contains an `AuthnStatement` and an `AttributeStatement`, which contains the requested attributes (cf. Section 4.4.2.2).

4.4.3 Identity Provider inside the eID-Token

Against the background of the novel concepts introduced above one may even envision a SAML-based Identity Provider, which is directly realized inside the eID-Token.

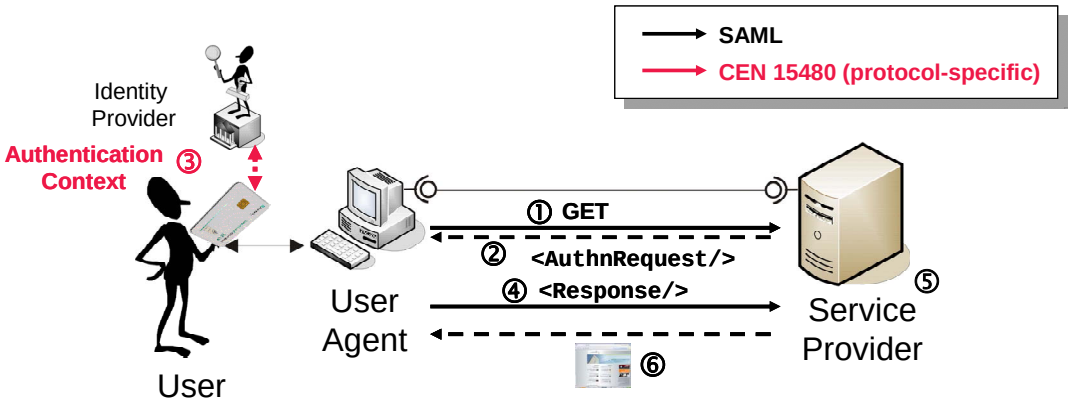

Figure 4: Scenario with IdP in eID

5 Conclusion

In this paper we have introduced a new smart card interface: The Web Service based Service Access Layer defined in **[ISO/IEC 24727]**. Since this interface builds upon existing well known and widely used Internet protocols it will facilitate a much better interoperability.

Using this as a basis, we have shown how the Extended Access Control Protocol can be implemented using this interface and how “secure messaging”-like trusted channels may be implemented in this case.

Additionally, we have discussed different options for the integration of current and future eID-tokens in SAML-infrastructures and in particular sketched a SAML-profile, which integrates the recent eID-standards **[ISO/IEC 24727]** and **[CEN 15480]**, optimizes the message flow, avoids security problems and even allows the realisation of an Identity Provider directly on a smart card.

References

[BioP@ss] The BioP@ss homepage: www.biopass.eu

[CEN 15480] Comité européen de normalisation (CEN): *Identification card systems — European Citizen Card — Part 1-4*, Technical Standard (partly in preparation), 2010

[EHS09] J. Eichholz, D. Hühnlein, J. Schwenk: *SAMLizing the European Citizen Card*, in A. Brömme & al. (Ed.), Proceedings of *BIOSIG 2009: Biometrics and Electronic Signatures*, GI-Edition Lecture Notes in Informatics (LNI) 155, 2009, pp. 105-117, http://www.ecsec.de/pub/SAMLizing-ECC.pdf

[ISO/IEC 7816] ISO/IEC: Identification cards – Integrated Circuit Cards, Part 1-13 & 15, International Standard

[ISO/IEC 24727] ISO/IEC: Identification Cards — Integrated Circuit Cards Programming Interfaces — Part 1-6, International Standard (partly in preparation), 2010

[JC30] Java Card™ Platform, Version 3.0 Connected Edition, http://java.sun.com

[LHP02] H. Leitold, A. Hollosi, R. Posch: *Security Architecture of the Austrian Citizen Card Concept*, Proceedings of the 18th Annual Computer Security Applications Conference, IEEE Press, 2002, pp. 391-401

[SAML(v2.0)] S. Cantor, J. Kemp, R. Philpott, E. 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

[SAML-Auth(v2.0)] J. Kemp, S. Cantor, P. Mishra, R. Philpott, E. 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)] S. Cantor, F. Hirsch, J. Kemp, R. Philpott, E. 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-HoK] 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-Prof(v2.0)] S. Cantor, J. Kemp, R. Philpott, E. 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.

[STORK] ***S**ecure iden**T**ity acr**O**ss bo**R**ders lin**K**ed (STORK) project website*, http://www.eid-stork.eu, 2010

[STORK-D.5.8.1b] J. Alcalde-Moraño, J. L. Hernández-Ardieta, A. Johnston, D. Martinez, B. Zwattendorfer: *STORK Deliverable D5.8.1b – Interface Specification*, 08.09.2009, https://www.eid-

stork.eu/index.php?option=com_processes&Itemid=&act=streamDocument&did=960

[TR-03110(V2.02)] Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik, BSI): 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 2.02, https://www.bsi.bund.de/cae/servlet/contentblob/532066/publicationFile/44802/TR-03110_v202_pdf.pdf , 2009.

[TR-03112(v1.1)] Federal Office for Information Security (Bundesamt für Sicherheit in der Informationstechnik, BSI): *Technical Directive eCard-API-Framework*, Version 1.1 of 15.07.2009, https://www.bsi.bund.de/cln_156/sid_BFE35DE615DDE059B55587F30981D6BD/ContentBSI/Publikationen/TechnischeRichtlinien/tr03112/index_htm.html

[WS-SecCon(v1.4)] A. Nadalin, M. Goodner, M. Gudgin, A. Barbir, H. Granqvist: *WS-SecureConversation 1.4*, OASIS Standard http://docs.oasis-open.org/ws-sx/ws-secureconversation/v1.4/ws-secureconversation.pdf, 2009

[XMLEnc] XML Encryption Syntax and Processing, www.w3.org

[XMLSig] XML Signature Syntax and Processing, www.w3.org