Innovative Building Blocks for Versatile Authentication within the SkIDentity Service

Detlef Hühnlein[1] Max Tuengerthal[1] Tobias Wich[1] Tina Hühnlein[1] Benedikt Biallowons[1]

Abstract: Accepting arbitrary electronic identity cards (eIDs) and similar authenticators in cloud and web applications has been a challenging task. Thanks to the multiply awarded "SkIDentity Service" this has changed recently. This versatile authentication infrastructure combines open technologies, international eID standards and latest research results with respect to trusted cloud computing in order to offer electronic identification and strong authentication in form of a trustworthy, simple to use and cost efficient cloud computing service, which supports various European eIDs as well as alternative authenticators proposed by the FIDO Alliance for example. The present contribution exposes innovative and patent pending building blocks of the SkIDentity Service: (1) The "Identity Broker", which eases the integration of authentication, authorization, federation and application services and in particular allows to derive secure credentials from conventional eID cards, which can be transferred to mobile devices for example. (2) The "Universal Authentication Service" (UAS), which allows to execute arbitrary authentication protocols, which are specified by the recently introduced "Authentication Protocol Specification" (APS) language, (3) the "Cloud Connector" which eases the integration of federation protocols into web applications and last but not least (4) the "SkIDentity Self-Service Portal", which makes it extremely easy for Service Providers to configure the necessary parameters in order to connect with the SkIDentity Service and use strong authentication in their individual applications.

1 Introduction

As the inherent weaknesses of password-based authentication [Ne94, IWS04] are about to become obvious in practice (see [Fe14a, Fe14b, CN14] for example) there seems to be a trend towards implementing strong authentication for web-based applications [Go11, Am13b, Mi13, Li13, FI] using a variety of protocols and authentication means. While supporting versatile authentication technologies certainly promotes the diffusion and adoption in practice [HRZ10], it also imposes the new challenge how to integrate and handle the large variety of involved technologies in an efficient manner. A basic strategy for handling this kind of complexity is to introduce appropriate interfaces, which allow to decouple certain services and modules, which can be developed, maintained and integrated in an independent manner. On a macro scale this approach has lead to the versatile authentication infrastructure designed and developed within the SkIDentity project, which has been supported by the German government within the "Trusted Cloud"[2] programme (see Section 2

[1] ecsec GmbH, Sudetenstraße 16, 96247 Michelau, Germany, {firstname.secondname}@ecsec.de

[2] See `http://trusted-cloud.de`.

This work was partly supported by the German Federal Ministry of Economics and Energy (BMWi) within the Trusted Cloud project **SkIDentity** (01MD11025A) (see `https://skidentity.com`) and the European Union within the 7th FP project **FutureID** (ICT-318424) (see `https://futureid.eu`).

and especially Figure 1) and on a micro scale to the highly modular and extensible Open eCard App (see [Wi13]), which allows to support arbitrary smart cards and authentication protocols in an efficient manner.

Against this background we will go one step further here and expose some innovative and patent pending building blocks of this versatile authentication system in Section 3: (1) The "Identity Broker" (see Section 3.1), which allows to integrate arbitrary services for authentication, authorization and federation and in particular allows to derive secure credentials from conventional eID cards, which can be transferred to mobile devices for example. (2) The "Universal Authentication Service" (UAS) (see Section 3.2), which allows to execute arbitrary authentication protocols, which are specified by the recently introduced "Authentication Protocol Specification" (APS) language [AM13a, AM15]. (3) The "Cloud Connector" (see Section 3.3) which eases the integration of federation protocols into web applications and last but not least (4) the "SkIDentity Self-Service Portal" (see Section 3.4), which makes it extremely easy for Service Providers to configure the necessary parameters in order to integrate with the SkIDentity Service in order to use strong authentication. Section 4 summarizes the main aspects of the present contribution and provides an outlook towards future developments.

2 Overview of the SkIDentity system

The main contribution of the present paper is to expose some innovative and patent pending building blocks of the SkIDentity system as outlined in [Sk12] and Figure 1. For this purpose we start by briefly recalling the main aspects of the SkIDentity Reference Architecture.

The SkIDentity system is depicted in Figure 1 and builds upon the concept of Federated Identity Management as explained in [MR08, HRZ10, Ca05a]. It refines the classical components "Client", "Service Provider" and "Identity Provider" in order to support arbitrary authentication mechanisms, eID-tokens, credential technologies and federation protocols.

There are components at the Client, the Service Provider and within the SkIDentity Service.

2.1 System Components at the Client

The system of the User (Client) comprises the *User Agent* (UA), which can be realised by an arbitrary browser, and an appropriate *eCard-App* (eCA), such as the Open eCard App [Hü12, Wi13], which enables the User to authenticate at an *Authentication Service* (AS) using some Credential. Due to the modular architecture based on ISO/IEC 24727 [IS08] it is easy to support various smart cards and authentication protocols. Using the add-on framework introduced in [Wi13] it is also easy to add application-specific logic[3], which can be accessed via corresponding interfaces.

[3] See [Ku13] for an example.

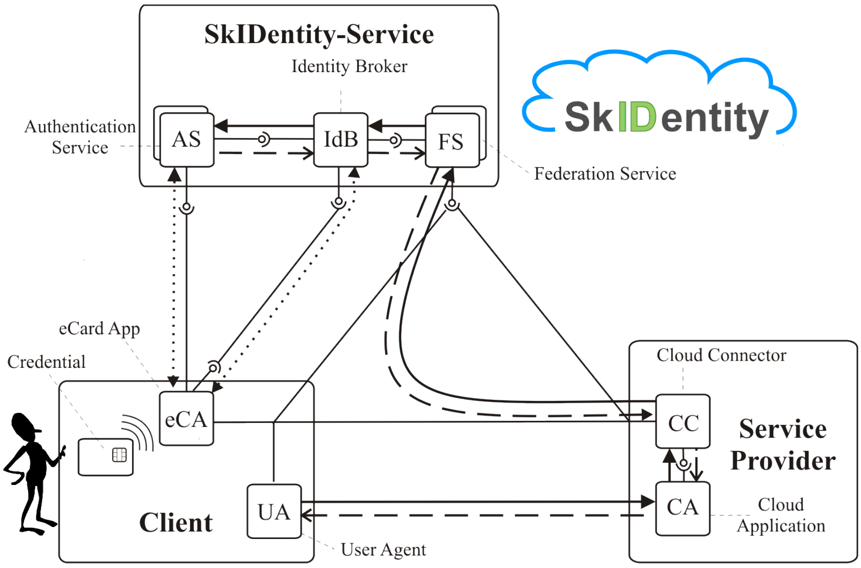


Fig. 1: SkIDentity Reference Architecture

2.2 System Components at the Service Provider

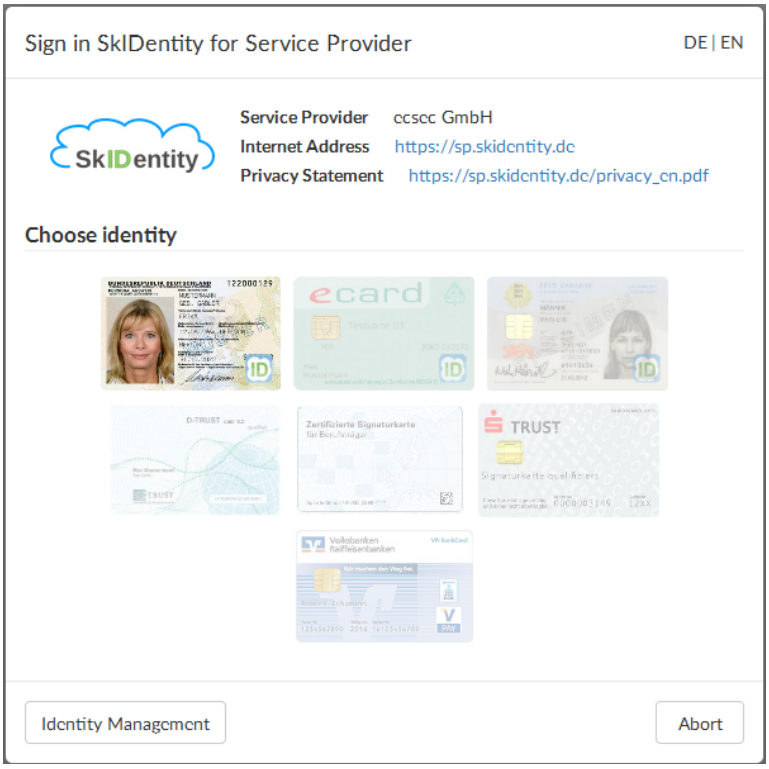
The system of the Service Provider (SP) comprises the *Cloud Application* (CA) and an appropriate *Cloud Connector* (CC) (see Section 3.3), which allows to communicate with an appropriate *Federation Service* (FS) in the SkIDentity Infrastructure using an appropriate federation protocol such as SAML [Ca05a], OpenID [Op] or OAuth [HL10, Ha12, HJ12] for example.

2.3 SkIDentity Service

Within the SkIDentity Service there are various *Federation Services* (FS) and a variety of *Authentication Services* (AS), which are connected via the *Identity Broker* (IdB) (see Section 3.1). The Identity Broker acts as information intermediary and provides the different eID-Services in a bundled and rehashed way. This offers the possibility to use the different services and tokens (electronic identity cards, electronic health cards, health professional cards, bank and signature cards, company ID tokens and last but not least FIDO's U2F token [FI]) with an easy and consistent interface for the secure authentication in cloud based applications.

3 Innovative Building Blocks of the SkIDentity Service

This section exposes innovative and patent pending building blocks of the SkIDentity Service. The "Identity Broker" is discussed in Section 3.1, the "Universal Authentication Service" is subject of Section 3.2, the "Cloud Connector" is subject of Section 3.3 and the "SkIDentity Self-Service Portal" finally is introduced in Section 3.4.



Sign in SkIDentity for Service Provider

Service Provider ecsec GmbH

Internet Address https://sp.skidentity.de

Privacy Statement https://sp.skidentity.de/privacy_en.pdf

Choose identity

Fig. 2: Identity Selector within the Identity Broker

3.1 Identity Broker

As depicted in Figure 1 the Identity Broker (IdB) is the central component within the SkIDentity Service, which receives authentication requests from some FS and forwards this request to an appropriate AS. This service performs the authentication of the User and returns the result to the IdB, which will return the received data to the calling FS. The selection of the AS is performed based on (1) the authentication options acceptable by the Service Provider, (2) the technical capabilities of the Client (e.g. whether an eID client software is present or not) and finally (3) the credential selected by the User among the possible options as depicted in Figure 2. Based on this information the IdB is able to determine a suitable AS, which will perform the authentication of the User.

The set of acceptable authentication options and requested attributes is specified by the Service Provider using the SkIDentity Self-Service Portal (see Section 3.4), which translates the choices to corresponding XML-based SAML Metadata structures, as outlined in [HTW14].

The IdB may not only act as a dispatcher, which simply forwards messages to some AS, but the IdB may also initiate the derivation of a cryptographically protected credential from a conventional eID card. Such a "Cloud Identity" can be securely stored on the User's system, transferred to another device of the User (e.g. his personal smart phone) and it may be bound to an additional cryptographic hardware token, in order to enhance security. A Cloud Identity may be seen as a cryptographically secured copy of an original eID, which may substitute a real eID in various online scenarios, while supporting a high level of usability. As a Cloud Identity can be transferred to arbitrary smart phones, this approach turns SkIDentity into a "Mobile eID as a Service" platform.

3.2 Universal Authentication Service

The Universal Authentication Service (UAS) is a specifically powerful Authentication Service, which has been developed within the FutureID project and which makes it easy to support arbitrary authentication protocols.

As the existing eID cards, eHealth cards, and eSign cards already support a large variety of different authentication protocols and it is expected that future authentication tokens will support other credentials and authentication protocols, it would be close to impossible to implement all required protocols using a conventional approach, because this would require a specialized program module for each authentication protocol.

In order to solve this problem, protocols are described in the Authentication Protocol Specification (APS) language [AM13a, AM15]. The APS descriptions of the authentication protocols in turn refer to appropriate Basic Services, such as cryptographic primitives or smart card commands according to ISO/IEC 7816 [IS]. As the different authentication protocols are all composed of a rather limited set of Basic Services, the problem of supporting arbitrary authentication protocols is reduced to providing this limited set of basic functionality and providing appropriate APS descriptions for the different authentication protocols.

Another advantage of specifying authentication protocols in the APS language is that the APS language is directly supported by state of the art formal protocol analysis tools, such as OFMC [MV09], that can be used to prove security properties of the authentication protocols.

The core component of the UAS is the Job Execution Environment (JEE) (see Section 3.2.2), which runs authentication protocols specified in the APS language. This makes it possible to support arbitrary protocols in a very efficient manner.

3.2.1 Authentication Protocol Specification Language

Describing the APS language is beyond the scope of this paper and we refer to [AM13a, AM15] for details. Instead, in Listing 1, we present an example which demonstrates the flavor of describing authentication protocols in APS. In this protocol, two parties (PCD and PICC) want to authenticate each other using an authenticated Diffie-Hellman key exchange protocol. The specification in Listing 1 consists of (1) the protocol name, (2) type declarations, (3) message formats, (4) the initial knowledge of the participants, (5) the actions that describe the message that are sent/received by the parties (this is the main part of the specification), and (6) the goals (security properties) this protocol must satisfy.

```
Protocol: EAC
Types:
  Nonce RpiccTA, RpiccCA;
  ...
Formats:
  eac1input( Msg, ImpData, ImpData, ImpData, ImpData );
  eac1output( ImpData, ImpData, ImpData, efcardaccess, Agent, Nonce );
  ...
Knowledge:
  PCD:  cert(PCD,pk(PCD),ca), pk(PCD), pk(ca), ...;
  PICC: cert(PICC,exp(g,sk(PICC)),ca), pk(ca), sk(PICC), ...;

Actions:
  [PCD]*->*[PICC]: eac1input( cert(PCD,pk(PCD),ca), CertDesc, ... )
  [PICC]*->*[PCD]: eac1output( RC, CHAT, CAR, EFCA, IDPICC, RpiccTA )
  let PK_PCD = exp(g,X)
  let S_PCD  = sign(inv(pk(PCD)),(IDPICC,RpiccTA,comp(PK_PCD)))
  [PCD]*->*[PICC]: eac2input( CertChain, PK_PCD, S_PCD )
  let PK_PICC = exp(g,sk(PICC))
  let K       = exp(PK_PCD,sk(PICC))    # = exp(PK_PICC,X)
  let Kmac    = kdf(K,RpiccCA)
  let Tpicc   = mac(Kmac,PK_PCD)
  [PICC]*->*[PCD]: eac2output( cert(PICC,PK_PICC,ca), Tpicc, RpiccCA )

Goals:
  PICC authenticates PCD on Tpicc
  PCD authenticates PICC on Tpicc
  K secret of PICC, PCD
```

List. 1: The EAC protocol specified in the APS language.

3.2.2 Job Execution Environment

The Job Execution Environment (JEE) is able to load and execute protocols that are defined in the APS language. Since the JEE is not able to execute the abstract APS directly, it must first be compiled to some kind of executable script code. For this purpose the JEE supports JavaScript to which an APS file is compiled to be executed.

An important feature of the JEE is the possibility to access and execute the Basic Services (BS) which provide different functions for common tasks, e.g. to compute a hash value, obtain the status of a certificate via Online Certificate Status Protocol (OCSP) or to create a certain Application Protocol Data Unit (APDU), which is to be sent to an Interface Device (IFD) component, which in turn communicates with a smart card. Furthermore

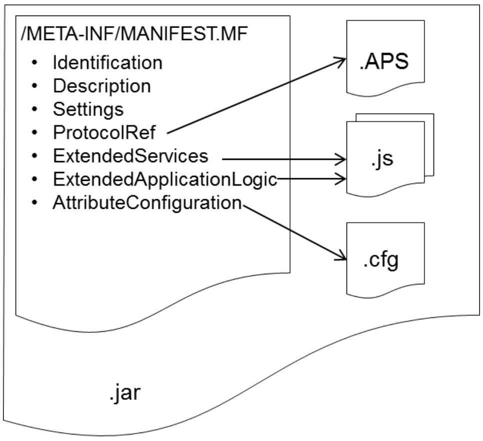


Fig. 3: Structure of a Credential-specific APS (CAPS) package.

the JEE may be equipped with additional JavaScript-based Extended Services (ES), which combine several calls and hence may be used to provide higher level functionality.

The difference between the Extended Services and the Basic Services is that the Basic Services are available in the Universal Authentication Service per default. Extended Services are usually more light-weight and are shipped together with a specific APS file. The environment that is needed by a specific protocol can be specified within the manifest file, which is distributed together with the APS file, which specifies the authentication protocol. The JEE uses this configuration file to set up a context in which the authentication protocol is executed. Every instance of a protocol has its own context so that the different instances do not interfere with each other.

Protocol descriptions are distributed in Java Archive (JAR) files that can be loaded by the Job Execution Environment during runtime. Since the leading factor when determining the protocol (and, hence, the JAR file) to be used for authentication is the type of the credential (i.e., the type of an eID card or some other authentication token) that is used for authentication, we call these JAR files Credential-specific APS (CAPS) packages. Besides the script files that define the protocol and the configuration that is used to set up the context for the protocol, a CAPS package can optionally contain additional Extended Services, which can be provided in form of JavaScript files. Furthermore, it optionally contains information about extended application logic that is to be executed after the authentication has been performed. For example, in case the credential is an eID card, the application logic might communicate with the card to sign messages or to obtain attributes from the card. Obtaining attributes using secure messaging established during authentication is the most common use case. It is therefore possible to provide an attribute configuration which provides information about how attributes are obtained and extracted from eID cards. The structure of a CAPS package is depicted in Figure 3.

We now describe the processing within the JEE in more detail. It can be structured into three phases: (1) Initialization, (2) Execution of the Authentication Protocol and (3) Execution of the Application Logic.

Initialization. In the initialisation phase, the JEE loads the CAPS package, creates an initial (job execution) context, for the protocol to be executed, and compiles the authentication protocol that is specified in the APS language into executable JavaScript code.

Execution of the authentication protocol. In this phase, the JEE first executes initialization code (if provided) and then executes the authentication protocol, i.e., the previously generated JavaScript code. During this execution, the JavaScript code may call predefined functions (crypt, decrypt, hash, etc.) to perform basic cryptographic operations. The JEE translates these function calls into calls to corresponding Basic Services (BS) or Extended Services (ES). Which algorithm to call (e.g., SHA-256 for hashing) is determined by the JEE during runtime using the settings given in the CAPS package (it may depend on the context, in particularly on messages received from the client). To generate and parse messages that are sent to/received from the client, the JavaScript code may use the message format objects that are provided by the CAPS package. Furthermore, the JavaScript code may call functions to send and receive messages to/from the network.

3.3 Cloud Connector

If the Cloud Application already supports a standardized federation protocol, such as SAML [Ca05a] or OAuth [HJ12] for example, it can directly communicate with the corresponding Federation Service within the SkIDentity Service. If not, it may perform the integration using the Cloud Connector (CC).

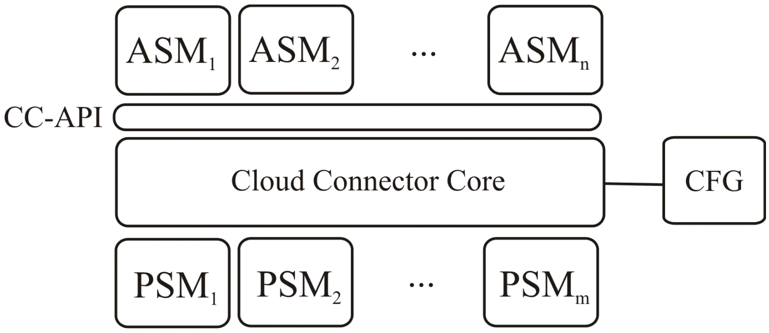


Fig. 4: Architecture of the SkIDentity Cloud Connector

As depicted in Figure 4, the CC is a modular integration library, which is available for different platforms, such as Java, PHP or .NET for example, and consists of a central component (Cloud Connector Core), which is accessible via a simple CC-API, which allows to

- request the authentication of the User (`authenticate( )`),
- get the identifier of the User determined during authentication (`getNameId( )`),
- access a particular attribute (`getAttribute($name)`) or all attributes of the User (`getAttributes( )`) or
- logout and redirect the User to a particular URL (`logout($return)`).

While the Platform Specific Modules (PSM$_i$) implement the different federation protocols (SAML, OAuth etc.), the Application Specific Modules (ASM$_j$) take care about the final integration into some application. There are various Application Specific Modules for popular Open Source applications, including Joomla, WordPress, ownCloud, MediaWiki, TYPO3, phpBB and Magento for example.

3.4 SkIDentity Self-Service Portal

While the integration of eIDs into cloud and web applications has been a challenging task, the SkIDentity Self-Service Portal[4] makes it easy for Service Providers to configure the parameters, which are required for the smooth integration of an individual service. The configuration can simply be performed by a responsive web application, which allows to specify (1) the information which is displayed to the User (see Figure 2), (2) the acceptable credentials and required attributes and (3) the corresponding technical parameters required for the federation protocol. As standardized SAML Metadata structures according to [Ca05b, Ca12] are used for this purpose, it is easy to import existing SAML Metadata files and export the generated data to another standardized system.

4 Summary and Outlook

The present paper exposed some innovative and patent pending building blocks of the multiply awarded SkIDentity Service, which makes it easy to accept eID cards and similar authenticators in cloud and web applications. In particular it was shown above that this system comprises an Identity Broker (see Section 3.1) which makes it easy to integrate arbitrary services for authentication and federation and create cryptographically protected derived credentials, which can be securely transferred to mobile devices for example. This gives rise to an innovative "Mobile eID as a Service" offering. Using the innovative Universal Authentication Service (see Section 3.2) one can support arbitrary authentication protocols, which are described by an appropriate "Alice and Bob"-like language as outlined in Listing 1. Last but not least it is easy to integrate cloud and web applications with the SkIDentity Service by using the convenient Self-Service Portal (see Section 3.4) and an appropriate Cloud Connector (see Section 3.3), if necessary.

[4] See `https://sp.skidentity.de`.

While the current focus of the SkIDentity Service is to provide strong authentication, future developments will extend the service in order to support authorization and provisioning as well as electronic signatures and in the long term perspective also the management of more complete business processes.

References

[AM13a] Almousa, Omar; Mödersheim, Sebastian: , Future AnB: The projected APS Language of FutureID. FutureID – WP42 / D42.3, 2013. http://futureid.eu/data/deliverables/year1/Public/FutureID_D42.03_WP42_v1.0_Design%20of%20formal%20APS-language.pdf.

[Am13b] Amazon Inc.: , AWS Multi-Factor Authentication, 2013. http://aws.amazon.com/de/mfa/.

[AM15] Almousa, Omar; Mödersheim, Sebastian: , Alice and Bob: Reconciling Formal Models and Implementation. submitted for publication, 2015. http://www.imm.dtu.dk/~samo/SPS.pdf.

[Ca05a] Cantor, Scott; Kemp, John; Philpott, Rob; Maler, Eve: , 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.

[Ca05b] Cantor, Scott; Moreh, Jahan; Philpott, Rob; Maler, Eve: , 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.

[Ca12] Cantor, Scott: , SAML V2.0 Metadata Extensions for Login and Discovery User Interface Version 1.0. OASIS Committee Specification 01, 2012. http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-metadata-ui/v1.0/sstc-saml-metadata-ui-v1.0.pdf.

[CN14] CNET: , eBay hacked, requests all users change passwords. Press Release 21.05.2014, 2014. http://www.cnet.com/news/ebay-hacked-requests-all-users-change-passwords/.

[Fe14a] Federal Office for Information Security: , Million-fold Identity Theft: Federal Office for Information Security offers security test for email addresses. Press Release 21.01.2014, in German, 2014. https://www.bsi.bund.de/DE/Presse/Pressemitteilungen/Presse2014/Mailtest_21012014.html.

[Fe14b] Federal Office for Information Security: , New Case of large-scale Identity Theft: Federal Office for Information Security informs victims. Press Release 07.04.2014, in German, 2014. https://www.bsi.bund.de/DE/Presse/Pressemitteilungen/Presse2014/Neuer_Fall_von_Identitaetsdiebstahl_07042014.htmll.

[FI] FIDO Alliance: , FIDO Alliance Specifications (UAF and U2F). https://fidoalliance.org/specifications/download/.

[Go11] Google: , Advanced sign-in security for your Google account, 2011. http://googleblog.blogspot.de/2011/02/advanced-sign-in-security-for-your.html.

[Ha12] Hardt, D.: , The OAuth 2.0 Authorization Framework. Request For Comments – RFC 6749, 2012. http://www.ietf.org/rfc/rfc6749.txt.

[HJ12] Hardt, D.; Jones, M.: , The OAuth 2.0 Authorization Framework: Bearer Token Usage. Request For Comments – RFC 6750, 2012. http://www.ietf.org/rfc/rfc6750.txt.

[HL10] Hammer-Lahav, E.: , The OAuth 1.0 Protocol. Request For Comments – RFC 5849, April 2010. http://www.ietf.org/rfc/rfc5849.txt.

[HRZ10] Hühnlein, Detlef; Rossnagel, Heiko; Zibuschka, Jan: Diffusion of Federated Identity Management. In: Tagungsband "Sicherheit 2010". volume 170 of LNI. GI, pp. 25–37, 2010. http://www.ecsec.de/pub/Sicherheit2010.pdf.

[HTW14] Horsch, Moritz; Tuengerthal, Max; Wich, Tobias: SAML Privacy-Enhancing Profile. In (Hühnlein, Detlef; Rossnagel, Heiko, eds): Proceedings of Open Identity Summit 2014. volume 237 of LNI. GI, pp. 11–22, 2014.

[Hü12] Hühnlein, Detlef; Petrautzki, Dirk; Schmölz, Johannes; Wich, Tobias; Horsch, Moritz; Wieland, Thomas; Eichholz, Jan; Wiesmaier, Alexander; Braun, Johannes; Feldmann, Florian; Potzernheim, Simon; Schwenk, Jörg; Kahlo, Christian; Kühne, Andreas; Veit, Heiko: On the design and implementation of the Open eCard App. In: Sicherheit 2012. GI-LNI, 2012. http://subs.emis.de/LNI/Proceedings/Proceedings195/95.pdf.

[IS] ISO/IEC 7816: , Identification cards – Integrated circuit cards – Part 1-15. International Standard.

[IS08] ISO/IEC: , ISO/IEC 24727: Identification cards – Integrated circuit cards programming interfaces – Part 1-6, 2008.

[IWS04] Ives, Blake; Walsh, Kenneth R; Schneider, Helmut: The domino effect of password reuse. Communications of the ACM, 47(4):75–78, 2004.

[Ku13] Kuhlisch, Raik; Petrautzki, Dirk; Schmölz, Johannes; Kraufmann, Ben; Thiemer, Florian; Wich, Tobias; Hühnlein, Detlef; Wieland, Thomas: An Open eCard Plug-in for accessing the German national Personal Health Record. In: Open Identity Summit 2013. volume 223 of GI-LNI, 2013.

[Li13] Lindemann, Rolf: Not Built On Sand – How Modern Authentication Complements Federation. In: Proceedings of Open Identity Summit 2013. volume 223 of Lecture Notes in Informatics. GI e.V., pp. 164–168, 2013.

[Mi13] Microsoft Inc.: , Microsoft Account Gets More Secure, 2013. http://blogs.technet.com/b/microsoft_blog/archive/2013/04/17/microsoft-account-gets-more-secure.aspx.

[MR08] Maler, Eve; Reed, Drummond: The Venn of Identity: Options and Issues in Federated Identity Management. IEEE Security & Privacy Magazine, 6(2):16–23, 2008.

[MV09] Mödersheim, Sebastian; Viganò, Luca: The Open-Source Fixed-Point Model Checker for Symbolic Analysis of Security Protocols. In (Aldini, Alessandro; Barthe, Gilles; Gorrieri, Roberto, eds): FOSAD. volume 5705 of Lecture Notes in Computer Science. Springer, pp. 166–194, 2009.

[Ne94] Neumann, Peter G.: Risks of passwords. Commun. ACM, 37(4):126, 1994.

[Op] OpenID Foundation: , OpenID Authentication 2.0. Final, December 5, 2007. `http://openid.net/specs/openid-authentication-2_0.html`.

[Sk12] SkIDentity-Team: , SkIDentity - Reference Architecture. Version 1.0, 2012.

[Wi13] Wich, Tobias; Horsch, Moritz; Petrautzki, Dirk; Schmölz, Johannes; Hühnlein, Detlef; Wieland, Thomas; Potzernheim, Simon: An extensible platform for eID, signatures and more. In: Proceedings of Open Identity Summit 2013. volume 223 of Lecture Notes in Informatics. GI e.V., pp. 55–68, 2013.