Starke, kosteneffiziente und benutzerfreundliche Authentisierung in der Cloud – ein Widerspruch in sich?

D. Hühnlein[1], J. Schmölz[1], T. Wich[1], B. Biallowons[1], T. Hühnlein[1]

Kurzfassung:
Für ein vertrauenswürdiges Cloud Computing werden zuverlässige Mechanismen für die starke, auf mindestens zwei Faktoren (Besitz, Wissen, Sein, etc.) basierende, Authentisierung benötigt [BSI-MSACC]. Damit ein Cloud-Anbieter nicht seinen potentiellen Kunden erst geeignete Authentisierungstoken zur Verfügung stellen muss, bietet es sich an, die zunehmend verfügbaren Chipkarten-basierten Ausweise, wie z.B. den neuen Personalausweis (nPA) oder die elektronische Gesundheitskarte (eGK), zur starken Authentisierung in der Cloud zu nutzen. Obwohl die Benutzerfreundlichkeit bekanntlich zu den wichtigsten Erfolgsfaktoren für die Akzeptanz von Authentisierungslösungen zählt [Senk12], zeichnen sich die bisher verfügbaren Lösungsansätze neben dem eingeschränkten Funktionsumfang leider auch durch eine ausgesprochen geringe Benutzerfreundlichkeit aus.
Vor diesem Hintergrund wird in diesem Beitrag zunächst die SkIDentity-Referenzarchitektur vorgestellt, durch die eine starke Authentisierung in der Cloud mit beliebigen Ausweiskarten ermöglicht wird. Auf dieser Grundlage werden Anforderungen für die Client-Komponente ermittelt und anhand der Betrachtung der bislang existierenden Komponenten [AusweisApp, bosAutent, AgetoApp] werden konkrete Vorschläge zur Verbesserung der Benutzerfreundlichkeit entwickelt, die in die Entwicklung der Open eCard App [HPS+12, HHB+13] einfließen und längerfristig hoffentlich auch in anderen eID-Clients ihren Niederschlag finden werden.

Stichworte: Authentisierung, eID, nPA, eGK, AusweisApp, Open eCard, Cloud Computing, Trusted Cloud, SkIDentity

1. Einleitung

Dem so genannten „Cloud Computing“ [BYV09, NIST-800-145, ShKa09], bei dem verschiedenste IT-Dienste bei Bedarf einfach „aus der Wolke“ bezogen werden können, wird eine große Zukunft vorausgesagt. Beispielsweise soll sich das deutsche Marktvolumen im Bereich der öffentlich angebotenen „Public Clouds“ von 702 Mio. € im Jahr 2010 bis zum Jahr 2025 auf 21,99 Mrd. € erhöhen und somit mehr als verdreißigfachen [Ber+10]. Auf der anderen Seite wurde in [SHJ+11] gezeigt, dass selbst die Cloud-Angebote der international führenden Anbieter angreifbar sind und gezielte Einbrüche in Cloud-Anwendungen [Robe12] zu signifikanten wirtschaftlichen Schäden führen können. Vor diesem Hintergrund ist es wenig verwunderlich, dass das Bundesamt für Sicherheit in der Informationstechnik (BSI) in [BSI-MSACC] generell für Administrationszugänge, sowie bei Cloud-Angeboten mit hohem Schutzbedarf, den Einsatz von starken, auf mindestens zwei Faktoren (Besitz, Wissen, Sein, etc.) basierenden Authentisierungsmechanismen fordert. Damit ein Cloud-Anbieter nicht seinen potentiellen Kunden in einem kostspieligen Prozess erst geeignete

[1] ecsec GmbH, Sudetenstr. 16, 96247 Michelau

Authentisierungstoken zur Verfügung stellen muss, wurde im SkIDentity-Projekt (siehe www.skidentity.de und [HHR+11]) vorgeschlagen, die verschiedenen Chipkarten der eCard-Strategie der Bundesregierung [Kowa07] und weitere bereits im Feld befindliche Authentisierungstoken zusammen mit den entsprechenden Authentisierungsdiensten (z.B. eID-Services für den neuen Personalausweis) für die starke Authentisierung in der Cloud zu nutzen. Wie in Abschnitt 2 und [HSW+12] erläutert, wird dies dadurch erreicht, dass ein „Identity Broker“ auf Basis der Anforderungen des Cloud-Dienstes und der beim Benutzer verfügbaren Komponenten und Ausweise einen geeigneten Dienst für die tatsächliche Durchführung des Authentisierungsvorganges auswählt. Daraus ergeben sich direkt die in Abschnitt 3 skizzierten Anforderungen an die Client-Komponente, die bei den bislang existierenden Komponenten [AusweisApp, bosAutent, AgetoApp] leider nicht bzw. nur ansatzweise umgesetzt sind. Vor diesem Hintergrund werden in Abschnitt 4 konkrete Verbesserungsvorschläge entwickelt, die zukünftig zu einer Steigerung der Benutzerfreundlichkeit in diesem Bereich beitragen sollen. In Abschnitt 5 werden schließlich die wichtigsten Aspekte dieses Beitrags zusammengefasst.

2. Die SkIDentity-Referenzarchitektur

Im Rahmen des SkIDentity-Projektes, das zu den Gewinnern des „Trusted Cloud“ Technologiewettbewerbs des Bundesministerium für Wirtschaft und Technologie (BMWi) zählt, wurde eine umfassende Referenzarchitektur für die starke Authentisierung in der Cloud entwickelt. Die in Abbildung 1 dargestellte und in [HSW+12, HHB+13] näher erläuterte Systemarchitektur umfasst Komponenten beim Benutzer, beim Diensteanbieter und in der SkIDentity-Infrastruktur:

- **Komponenten beim Benutzer**

  Das System des Benutzers (Client) umfasst einen User Agent (UA), der beispielsweise durch einen beliebigen Browser realisiert sein kann, und eine so genannte eCard App (eCA) (vgl. [AusweisApp, bosAutent, AgetoApp, HPS+12, HHB+13]), die unter Verwendung des digitalen Ausweises (Credential) des Benutzers (User) eine Authentisierung gegenüber dem Authentication Service (AS) in der Infrastruktur durchführt.

- **Komponenten beim Diensteanbieter**

  Das System des Diensteanbieters (Service Provider) umfasst die eigentliche Anwendung (Cloud Application (CA)) und einen so genannten Cloud Connector (CC), der beispielsweise über Schnittstellen wie [JAAS] in diese integriert sein kann und die Kommunikation mit dem Federation Service (FS) in der SkIDentity-Infrastruktur übernimmt.

- **SkIDentity-Infrastruktur**

  In der SkIDentity-Infrastruktur für die starke Authentisierung in der Cloud existieren Federation Services (FS) und Authentication Services (AS), die über einen Identity Broker (IdB) miteinander verbunden sind. Hierbei führt der AS die tatsächliche Authentisierung durch, während der FS die benötigte

Funktionalität für ein möglicherweise gewünschtes Single Sign-On bereitstellt und die hierfür vorgesehenen Föderationsprotokolle, wie z.B. [SAML(v2.0), OpenID(v2.0), RFC5849, RFC6749, ID-MI(v1.0)] unterstützt.

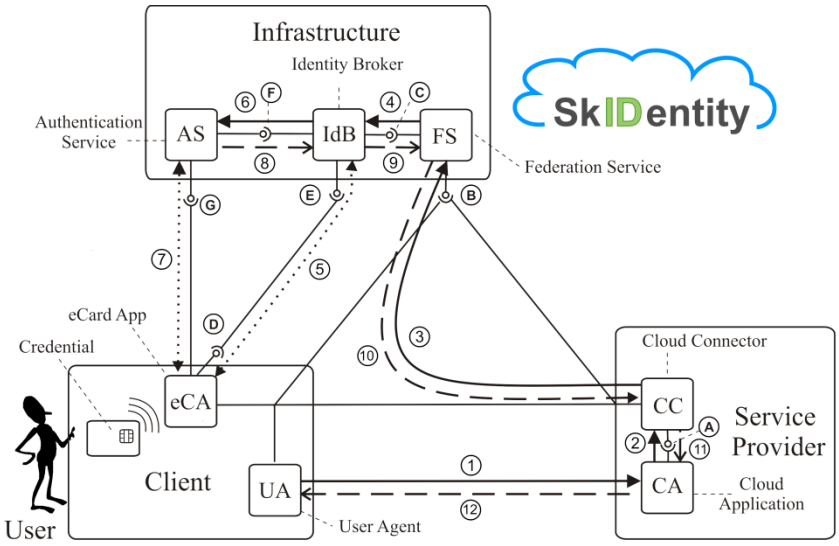

Abbildung 1: SkIDentity-Referenzarchitektur für die starke Authentisierung in der Cloud

Wie in [HSW+12] ausführlich erläutert, ermittelt der Identity Broker in Schritt (5) welche Komponenten und Ausweise beim Benutzer vorhanden sind, damit in den Schritten (6) und (7) ein geeigneter Authentisierungsdienst kontaktiert werden kann.

3. Anforderungen an eine benutzerfreundliche eCard App und ihre Erfüllung durch derzeit existierende Komponenten

Damit die starke Authentisierung in der Cloud in einer benutzerfreundlichen Art und Weise erfolgen kann, muss die eCard App unter anderem folgende Anforderungen erfüllen:

R1. Nutzbarkeit auf beliebigen Plattformen und mit beliebigen Browsern

Nichts ist weniger benutzerfreundlich, als eine Software, die nicht für die angestammte Plattform des Benutzers verfügbar ist, nicht zuverlässig funktioniert oder nicht in Verbindung mit den gewohnten Anwendungen (z.B. dem Lieblingsbrowser des Benutzers) verwendet werden kann. Deshalb soll die eCard App möglichst auf beliebigen Plattformen und in Verbindung mit beliebigen Browsern und sonstigen Anwendungen genutzt werden können.

Während die [AusweisApp] bislang nur für ausgewählte PC-Plattformen verfügbar ist und bislang über entsprechende PlugIn-Mechanismen den Internet Explorer sowie die Extended Support Releases (ESR) des Firefox unterstützt, sind die alternativen Komponenten durch den Java-basierten Ansatz weitgehend unabhängig von der Plattform und durch die Bereitstellung als Applet (siehe [bosAutent]) bzw. durch die alternative eID-Aktivierung über den auf localhost

verweisenden Link gemäß [BSI-TR-03112-7, Section 3.2.1] (siehe [AgetoApp]) unabhängig von den eingesetzten Browsern.

R2. Erkennung ob eCard App installiert und verfügbar ist

Damit bei Bedarf ein Download-Vorgang angeboten werden oder hilfsweise ein eCard-Applet ausgeliefert werden kann, muss es für die aufrufende Webapplikation erkennbar sein, ob eine eCard App installiert und verfügbar ist.

Leider wird einer Webapplikation bei der derzeit in [BSI-TR-03112-7, Section 3.2.1] vorgesehenen eID-Aktivierung keine Möglichkeit eröffnet, zu erkennen, ob eine eCard App installiert und verfügbar ist. Somit bleibt einer Webapplikation nur die optimistische – aber derzeit leider fast immer falsche – Annahme, dass beim Benutzer eine entsprechende eCard App verfügbar sein könnte. Sofern aber unter http://localhost:24727 keine aktive eCard App vorhanden ist, führt der geplante Start des Authentisierungsvorgangs zu einer Fehlermeldung, dass die angefragte Webseite nicht gefunden wurde. Diese Fehlermeldung kann vermutlich nur von Experten, die mit den Details der eID-Aktivierung gemäß [BSI-TR-03112-7, Section 3.2.1] vertraut sind, richtig interpretiert werden. Eine benutzerfreundliche Fehlerbehandlung ist mit den derzeit spezifizierten Mitteln leider nicht ohne Weiteres möglich.

R3. Unterstützung beliebiger Authentisierungstoken, -dienste und -protokolle

Da die Cloud nicht an nationalen Grenzen endet, müssen neben dem nPA auch andere europäische Ausweiskarten und Authentisierungsdienste unterstützt werden. Die Unterstützung beliebiger Authentisierungstoken sowie Authentisierungs- und Föderationsprotokolle wäre langfristig wünschenswert.

Leider unterstützten die eID-Clients [AusweisApp, bosAutent, AgetoApp] bislang ausschließlich den nPA, der im Falle der [AusweisApp] bereits beim Start des Authentisierungsvorgangs gesteckt sein muss, damit kein benutzerunfreundlicher Timeout droht.

R4. Benutzerfreundlicher Einwilligungsdialog

Während der Benutzerdialog für die Einwilligung zum Zugriff auf die im nPA gespeicherten Daten aus dem Blickwinkel des Datenschutz beinahe[2] als vorbildlich gelten kann, scheint hier hinsichtlich der Benutzerfreundlichkeit im Einwilligungsdialog möglicherweise grundsätzlich und vor allem bei den verfügbaren „Komfort-Chipkartenlesern“ noch Potenzial für Verbesserungen zu existieren.

[2] Da der Betrieb des eID-Servers im Regelfall nicht selbst vom Diensteanbieter gemäß § 2 Abs. 3 PAuswG, sondern von einem gemäß § 11 BDSG beauftragten Dienstleister erbracht wird und dies zu zusätzlichen Gefährdungen für die Daten des Betroffenen führen kann, wäre es wünschenswert, wenn dies dem Bürger gegenüber transparent gemacht werden würde.

4. Vorschläge zur Steigerung der Benutzerfreundlichkeit

Die in Abschnitt 3 aufgestellten Anforderungen werden in diesem Kapitel in konkrete Verbesserungsvorschläge zur Steigerung der Benutzerfreundlichkeit überführt, wobei Abschnitt 4.x der Anforderung R.x gewidmet ist.

4.1. Nutzbarkeit auf allen Plattformen und mit beliebigen Browsern

Um eine eCard App mit überschaubarem Aufwand für unterschiedliche Plattformen bereitzustellen, bietet sich eine Implementierung in Java an. Darüber hinaus kann durch die kürzlich eingeführte alternative eID-Aktivierung gemäß [BSI-TR-03112-7, Section 3.2.1] auf wartungsintensive Browser-Plugins verzichtet und die gewünschte Unabhängigkeit vom eingesetzten Browser erreicht werden.

4.2. Erkennung ob eCard App installiert und verfügbar ist

Die alternative eID-Aktivierung gemäß [BSI-TR-03112-7, Section 3.2.1] hebt die bislang bestehende Browserabhängigkeit auf und ist deshalb ein sehr wichtiger Schritt zur Steigerung der Benutzerfreundlichkeit. Zugleich erschwert sie aber die Erkennung, ob eine eCard App installiert und verfügbar ist. Um dieses Problem zu lösen, wird die Nutzung des mittlerweile von praktisch allen[3] modernen Browsern unterstützte Cross-Origin Resource Sharing-Mechanismus [CORS] vorgeschlagen, durch den eine Webseite mittels JavaScript erkennen kann, ob unter http://localhost:24727 eine eCard App vorhanden ist, welche Funktionen diese App unterstützt und in welchem Status sich diese App befindet.

In aktuellen Web-Browsern verhindert die Same Origin Policy (SOP) [RFC6454] das Nachladen von Ressourcen durch eingebettete Skripte (z.B. JavaScript) von fremden Webseiten. Das Nachladen von Ressourcen aus derselben Quelle – d.h. aus der Quelle, aus der auch das Skript stammt – ist hingegen problemlos möglich. Die SOP ist immer dann erfüllt, wenn bei der Quelladresse das verwendete Protokoll, der Hostname sowie der Port übereinstimmen.

In Abbildung 2 ist dargestellt, wie mittels JavaScript und CORS die grundsätzliche Existenz sowie der aktuelle Status einer unter localhost verfügbaren eCard App ermittelt werden kann. In Schritt (1) und (2) wird mit dem Besuch der Seite eines Diensteanbieters – im hier angegebenen Beispiel demo.skidentity.de - ein geeignetes JavaScript ausgeliefert, das in Schritt (3) durch einen `XMLHttpRequest` (AJAX) den `getStatus`-Aufruf an localhost schickt. Sofern unter `localhost` eine geeignete eCard App vorhanden ist, wird in Schritt (4) ein entsprechendes `Status`-Element an den Browser zurückgeliefert.

[3] Siehe http://caniuse.com/#feat=cors.

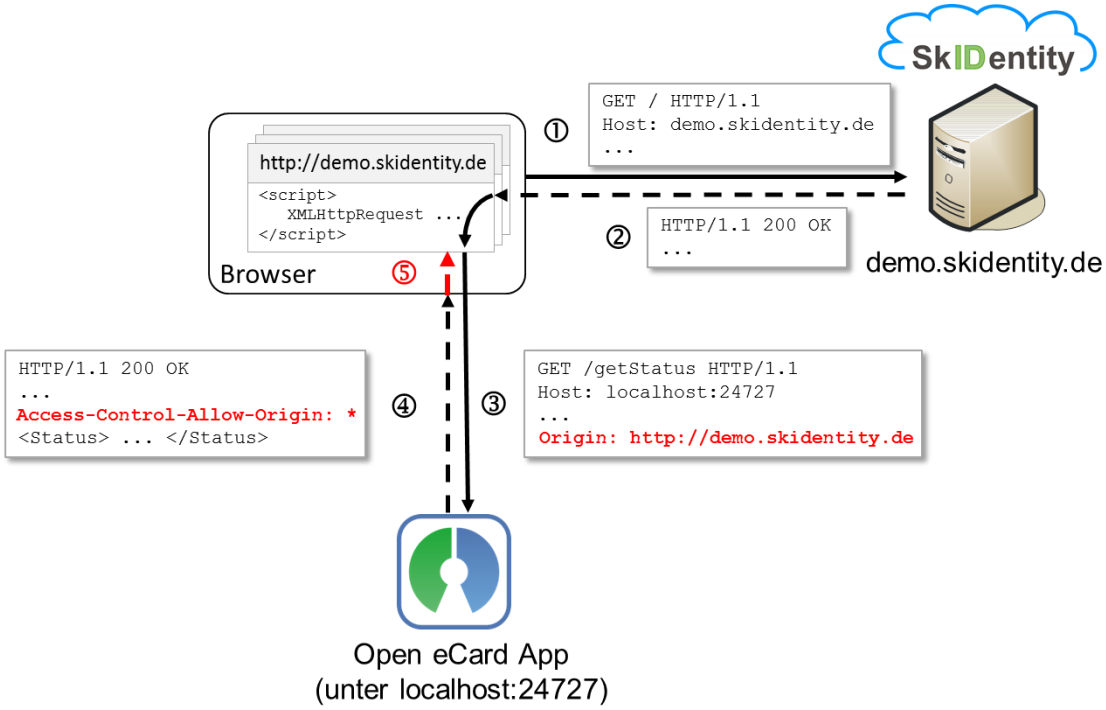

Abbildung 2: Ermittlung der Existenz und des Status einer eCard App auf localhost über CORS

Während das `Status`-Element zwar in Schritt (4) immer an den Browser zurückgeliefert wird, wird dieses – sofern nicht wie unten beschrieben in den Schritten (3) und (4) zusätzliche, CORS-spezifische http-Header eingefügt werden – gemäß der SOP *nicht* an das aufrufende JavaScript weitergereicht. Denn der Ursprung des JavaScripts, das im oben genannten Beispiel im Browser abläuft, ist demo.skidentity.de aber die von localhost:24727 zurückgelieferte Antwort hat einen anderen Ursprung und wird deshalb gemäß [RFC6454] nicht an das JavaScript im Browser ausgeliefert.

Glücklicherweise kann die SOP aber durch das Einfügen von sehr einfachen, in der [CORS]-Spezifikation definierten, http-Headern in Schritt (3) und Schritt (4) außer Kraft gesetzt werden, so dass das `Status`-Element tatsächlich dem aufrufenden JavaScript zur Verfügung gestellt wird und dort entsprechend ausgewertet werden kann.

In Schritt (3) wird ein zusätzlicher http-Header („`Origin`“) mitgeschickt, der angibt, dass ein Skript aus der angegeben Quelle (hier: http://demo.skidentity.de) eine Ressource nachladen möchte. Die Antwort in Schritt (4) enthält ebenfalls einen zusätzlichen http-Header („`Access-Control-Allow-Origin`“), der angibt, wer Ressourcen mittels CORS nachladen darf. Des Weiteren enthält diese Antwort auch gleich noch die angefragte Ressource in Form eines `Status`-Elementes. Sofern die Auswertung der Angabe im `Access-Control-Allow-Origin-Header` zu einem positiven Ergebnis führt, reicht der Browser die Ressource in Schritt (5) an das Skript weiter, ansonsten wird sie verworfen. Wie in Abbildung 2 dargestellt, kann durch die Verwendung der Wildcard „*“ im `Access-Control-Allow-Origin-Header`

klargestellt werden, dass die angefragten Status-Informationen Skripten aus beliebigen Quellen zur Verfügung gestellt werden darf.

Während der sicherheitstechnische Wert des CORS-Mechanismus und das zu Grunde liegende Angreifermodell sicherlich kritisch diskutiert werden könnte, muss hier festgestellt werden, dass durch die Verwendung von CORS in diesem Kontext keine nennenswerten zusätzlichen Risiken für die beteiligten Parteien (eCard App, Browser und Web-Applikation) entstehen. Da die Kommunikation in Schritt (3) und (4) unabhängig von der Verwendung des CORS-Mechanismus stattfindet, ändert sich die Bedrohungslage für die eCard App nicht wesentlich. Umgekehrt könnte eine nicht vertrauenswürdige eCard App über diesen Mechanismus statt dem angeforderten `Status`-Element entsprechend bösartiges JavaScript zurückliefern, das durch den CORS-Mechanismus im Browser zur Ausführung gebracht wird und die Webanwendung angreifen könnte. Da sich diese jedoch gut vor solchen Angriffen aus dem Browser schützen kann und im Fall einer nicht vertrauenswürdigen eCard App weitaus gravierendere Bedrohungen (z.B. Ausforschen und Missbrauch der eID-PIN) existieren, scheint mit der zusätzlichen Nutzung von CORS insgesamt keine nennenswerte Veränderung der Bedrohungslage verbunden zu sein.

Durch den CORS-Machanismus läßt sich also sehr benutzerfreundlich und ohne nennenswerte sicherheitstechnische Implikationen ermitteln, ob eine entsprechende eCard App verfügbar ist und welche Fähigkeiten diese besitzt. Somit können zukünftig die wenig aufschlussreichen Fehlermeldungen (z.B. „Fehler: Verbindung fehlgeschlagen“, die bei der alternativen eID-Aktivierung auf eine nicht vorhandene oder nicht gestartete eCard App zurückzuführen sind, vermieden und solche Fehlersituationen entsprechend benutzerfreundlich behandelt werden.

4.3. Unterstützung beliebiger Ausweistoken, -dienste und -protokolle

Für diese naheliegende aber schwierig zu realisierende Anforderung werden verschiedene Maßnahmen vorgeschlagen:

1) Ausweiskarten: CardInfo-Dateien und „Identity Selector“

Durch die in [CEN15480-3, ISO24727-3] standardisierte `CardInfo`-Struktur kann eine eCard App beliebige Ausweiskarten unterstützen, die durch eine solche XML-Struktur beschrieben sind und standardisierte Authentisierungsprotokolle (siehe z.B. [BSI-TR-03112-7]) nutzen. Wie beim Applet-basierten Demonstrator unter http://openecard.org/demo gezeigt, können die verfügbaren Kartenterminals sowie die gesteckten Karten dem Benutzer in einem so genannten „Identity Selector“ durch entsprechende Bilder angezeigt werden. Damit die hierbei in der eCard App durch das Hinzufügen, Erkennen und Entfernen von Kartenterminals und Karten entstehenden Ereignisse in einer Webseite oder einer anderen Applikation angezeigt werden können, wird vorgeschlagen, die localhost-Schnittstelle der eCard App um die Funktionen `getStatus` und `waitForChange` zu erweitern.

- `getStatus`

  Die Funktion `getStatus` liefert Informationen über die Fähigkeiten der eCard App sowie deren momentanen Zustand. Zu den Informationen zählen Name und Versionsnummer der eCard App, die unterstützten Versionen des eCard-API-Frameworks, die von der App unterstützten Karten sowie zugehörige Protokolle und etwaige weitere Fähigkeiten, die über das eCard-API-Framework hinausgehen. Zudem werden die momentan zur Verfügung stehenden Karten zurückgeliefert.

- `waitForChange`

  Die Funktion `waitForChange` liefert alle Änderungen, die sich seit dem Aufruf von `getStatus` bzw. dem letzten Aufruf von `waitForChange` ergeben haben.

2) Beliebige Authentisierungstoken und –dienste: Identity Broker

Durch den Identity Broker in der SkIDentity-Infrastruktur (siehe Abbildung 1) können beliebige Authentisierungsdienste genutzt werden. Darüber hinaus erscheint die Unterstützung der derzeit in der „Account Chooser“ Arbeitsgruppe der OpenID Foundation (siehe http://ac.openid.net/) entstehenden Spezifikationen im oben genannten „Identity Selector“ sehr vielversprechend, da hierdurch bereits existierende Identity Management Infrastrukturen integriert werden können.

3) Beliebige Authentisierungsprotokolle: Universeller Authentisierungsdienst

Damit schließlich beliebige Authentisierungsprotokolle unterstützt werden können, muss die eCard App bzw. ein „universeller Authentisierungsdienst“ zusätzlich in der Lage sein, beliebige, in einer formalen Sprache (siehe z.B. [AVISPA-D2.1]) spezifizierte, Authentisierungsprotokolle ausführen zu können. Die Machbarkeit dieses reizvollen Lösungsansatzes wird im Rahmen des von der EU geförderten FutureID-Projektes [RCF+12] untersucht.

4.4. Benutzerfreundlicher Einwilligungsdialog

Beispielsweise könnte die Anzeige zunächst auf die besonders relevanten Informationen beschränkt werden und weiterführende Details könnten in einem explizit „aufgeklappten Dialogfenster“ zugänglich gemacht werden. Um eine intuitive Benutzerführung zu ermöglichen, könnte das Logo des anfragenden Diensteanbieters in den Einwilligungsdialog integriert werden.

Bei Verwendung eines Standard- oder Komfort-Lesers bekommt der Nutzer die zur Übermittlung ausgewählten Daten bei der [AusweisApp] derzeit leider zweimal angezeigt. Zunächst im Einwilligungsdialog und ein zweites Mal auf dem Leser, wobei sie abermals durch den Nutzer bestätigt werden müssen. Dieser Ablauf ist nicht sehr benutzerfreundlich und sollte dahingehend optimiert werden, dass dem Benutzer

beim Einsatz eines Standard- oder Komfort-Lesers auf Wunsch die angeforderten Daten nur noch einmal angezeigt werden.

Außerdem wäre es denkbar, eine einmalige Einwilligung zur Übermittlung bestimmter Daten an einen Diensteanbieter, z.B. falls dieser seine Vertrauenswürdigkeit in geeigneter Weise nachgewiesen hat oder lediglich das Benutzer- und Dienstspezifische Pseudonym anfordert, in der eCard App zu speichern, damit hier keine erneute Einwilligung erfolgen muss. Da die Zwischenspeicherung einer zur Authentisierung genutzten PIN – anders als bei einer Signatur-PIN gemäß § 15 Abs. 2 Satz 1 Nr. 1 SigV – nicht explizit verboten ist, wäre es in besonders sicheren Einsatzumgebungen sogar denkbar, eine einmal eingegebene Authentisierungs-PIN in geeigneter Weise in der eCard App zwischenzuspeichern.

5. Zusammenfassung

In [BSI-MSACC] wird gefordert, dass beim Cloud Computing für Administratoren zwingend und bei hohem Schutzbedarf auch für Anwender starke, auf mindestens zwei Faktoren (Besitz, Wissen, Sein, etc.) basierende, Authentisierungsmechanismen eingesetzt werden. Vor diesem Hintergrund wurde in diesem Beitrag die SkIDentity-Referenzarchitektur vorgestellt, durch die eine starke Authentisierung in der Cloud mit beliebigen Ausweiskarten ermöglicht wird. Auf dieser Grundlage wurden die Anforderungen für die Client-Komponente ermittelt und ausgehend von den bislang existierenden Komponenten wurden konkrete Vorschläge zur Verbesserung der Benutzerfreundlichkeit entwickelt, die in die Entwicklung der Open eCard App [HPS+12, HHB+13] einfließen und längerfristig hoffentlich auch in anderen eID-Clients ihren Niederschlag finden werden.

Literatur

[AgetoApp] Ageto Innovation GmbH: AGETO AusweisApp, http://www.ageto.de/egovernment/ageto-ausweis-app, 2012

[AusweisApp] Bundesamt für Sicherheit in der Informationstechnik: Offizielles Portal für die „AusweisApp“, http://www.ausweisapp.de, 2012

[AVISPA-D2.1] AVISPA-Consortium: The High Level Protocol Specification Language, Deliverable D2.1, http://www.avispa-project.org/delivs/2.1/d2-1.pdf, August 2003

[Ber+10] Berlecon Research & al.: Das wirtschaftliche Potenzial des Internet der Dienste. Studie im Auftrag des Bundesministeriums für Wirtschaft und Technologie (BMWi). http://www.berlecon.de/idd, 2010

[bosAutent] bos GmbH & Co. KG: Governikus Autent, http://www.bos-bremen.de/de/governikus_autent/1854605/ , 2012

[BSI-MSACC] Bundesamt für Sicherheit in der Informationstechnik (BSI): Sicherheitsempfehlungen für Cloud Computing Anbieter – Mindestsicherheitsanforderungen in der Informationssicherheit, Eckpunktepapier, http://docs.ecsec.de/BSI-MSACC, 2011

[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 1.1.2, Part 7. http://docs.ecsec.de/BSI-TR-03112-7, 2012

[BYV09] R. Buyya, C. Yeo, S. Venugopal, J. Broberg, I. Brandic: Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems, 25(6):599–616, http://www.buyya.com/gridbus/papers/Cloud-FGCS2009.pdf, 2009

[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, 2010

[CORS] Anne van Kesteren: Cross-Origin Resource Sharing, W3C Working Draft 3 April 2012. http://www.w3.org/TR/cors/, April 2012

[HHB+13] Moritz Horsch , Detlef Hühnlein , Christian Breitenstrom , Thomas Wieland, Alexander Wiesmaier , Benedikt Biallowons, Dirk Petrautzki , Simon Potzernheim, Johannes Schmölz, Alexander Wesner, Tobias Wich: Die Open eCard App für mehr Transparenz, Vertrauen und Benutzerfreundlichkeit beim elektronischen Identitätsnachweis, 13. IT-Sicherheitskongress, 2013

[HHR+11] Detlef Hühnlein, Gerrit Hornung, Heiko Roßnagel, Johannes Schmölz, Tobias Wich, Jan Zibuschka: SkIDentity - Vertrauenswürdige Identitäten für die Cloud, DACH-Security 2011, 2011

[HPS+12] Detlef Hühnlein, Dirk Petrautzki, Johannes Schmölz, Tobias Wich, Moritz Horsch, Thomas Wieland, Jan Eichholz, Alexander Wiesmaier, Johannes Braun, Florian Feldmann, Simon Potzernheim, Jörg Schwenk, Christian Kahlo, Andreas Kühne, Heiko Veit: On the design and implementation of the Open eCard App, Sicherheit 2012, GI-LNI (2012), http://subs.emis.de/LNI/Proceedings/Proceedings195/95.pdf

[HSW+12] Detlef Hühnlein, Johannes Schmölz, Tobias Wich, Benedikt Biallowons, Moritz Horsch, Tina Hühnlein: Standards und Schnittstellen für das Identitätsmanagement in der Cloud, Tagungsband DACH Security 2012 (IT-Verlag, 2012), http://www.ecsec.de/pub/2012_DACH_IdM.pdf

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

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

[JAAS] SUN Inc.: Java Authentication and Authorization Service (JAAS). Reference Guide for the Java TM SE Development Kit 6. http://docs.oracle.com/javase/6/docs/technotes/guides/security/jaas/JAASRefGuide.html

[Kowa07] Bernd Kowalski: Die eCard-Strategie der Bundesregierung im Überblick. BIOSIG 2007: Biometrics and Electronic Signatures, 108 LNI, SS. 87–96, http://subs.emis.de/LNI/Proceedings/Proceedings108/gi-proc-108-008.pdf , 2007

[NIST-800-145] National Institute of Standards and Technology: The NIST Definition of Cloud Computing, NIST Special Publication 800-145. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[OpenID(v2.0)] OpenID Foundation: OpenID Authentication 2.0. Final, December 5, 2007. http://openid.net/specs/openid-authentication-2_0.html

[RCF+12] Heiko Roßnagel, Jan Camenisch, Lothar Fritsch, Thomas Groß, Detlef Houdeau, Detlef Hühnlein, Anja Lehmann, Jon Shamah: FutureID - Shaping the Future of Electronic Identity, Proceedings of Annual Privacy Forum 2012, LNCS (Springer, 2012)

[RFC5849] E. Hammer-Lahav: The OAuth 1.0 Protocol, Request For Comments – RFC 5849, http://www.ietf.org/rfc/rfc5849.txt, 2010

[RFC6454] A. Barth: The Web Origin Concept, Request For Comments – RFC 6454, http://www.ietf.org/rfc/rfc6454.txt, 2011

[RFC6749] D. Hardt: The OAuth 2.0 Authorization Framework, Request For Comments - RFC 6749, http://www.ietf.org/rfc/rfc6749.txt, 2012.

[Robe12] Paul Roberts: Cloud Service Linode Hacked, Bitcoin Accounts Emptied. Threat Post, 02.03.2012. http://docs.ecsec.de/Robe12, 2012.

[SAML(v2.0)] Scott Cantor, John Kemp, Rob Philpott, 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

[Senk12] Christian Senk: Future of Cloud-based Services for Multi-factor Authentication: Results of a Delphi Study. Proceedings of 3rd International Conference on Cloud Computing, CLOUDCOMP 2012, Vienna, Austria, September 24-26, 2012

[SHJ+11] Juraj Somorovski, Mario Heiderich, Meiko Jensen, Jörg Schwenk, Nils Gruschka, Luigi Lo Iacono: All your clouds are belong to us: security analysis of cloud management interfaces, Proceedings of the 3rd ACM workshop on Cloud computing security workshop, 2011, http://www.computerworld.com.pt/media/2011/10/AmazonSignatureWrapping.pdf

[ShKa09] S. Shankland, J. Kaden: Gartner: Cloud Computing wird wichtigster IT-Trend 2010, ZDNet-Beitrag, 21.10.2009, 2009, http://docs.ecsec.de/ShKa09