Elektronische Signaturen mit der Open eCard App

Detlef Hühnlein[1] · Johannes Schmölz[1]
Tobias Wich[1] · Andreas Kühne[2]

[1]ecsec GmbH
vorname.nachname@ecsec.de

[2]Trustable Ltd
kuehne@trustable.de

Zusammenfassung

Mit der eCard-Strategie der Bundesregierung, dem eCard-API-Framework und nicht zuletzt der darauf basierenden AusweisApp war die Hoffnung verbunden, dass eines Tages die starke Authentisierung und qualifizierte elektronische Signatur mit beliebigen Chipkarten leicht in beliebigen Anwendungen genutzt werden kann. Da die AusweisApp des Bundes aber diese Erwartungen nicht erfüllt hat, haben sich im Open eCard Projekt industrielle und akademische Experten zusammengeschlossen, um eine quelloffene und plattformunabhängige Implementierung des eCard-API-Frameworks und eine leichtgewichtige, vertrauenswürdige und benutzerfreundliche Alternative zur AusweisApp des Bundes – die so genannte Open eCard App – bereitzustellen. Der vorliegende Beitrag stellt das erweiterbare Design der Open eCard App vor und zeigt, wie auf dieser Basis Funktionen für die elektronische Signatur realisiert werden können.

1 Einleitung

Durch die elektronische Abwicklung von Geschäftsprozessen lassen sich Kosten senken sowie Fehlerquoten und Prozesslaufzeiten reduzieren. Dazu werden im Wesentlichen papiergebundene Schriftstücke zunehmend digitalisiert [BSI13a] oder durch elektronische Dokumente ersetzt und Prozesse automatisiert. Hierbei kann mittels elektronischer Signaturen [HüKo06] die Integrität und Authentizität der verarbeiteten Dokumente sichergestellt werden. Um Schriftformerfordernisse sowie besondere Sicherheitsanforderungen zu erfüllen oder um in den Genuss der Beweiserleichterung gemäß § 371a ZPO zu kommen, werden hierbei teilweise qualifizierte elektronische Signaturen (QES) gemäß § 2 Nr. 3 SigG eingesetzt.

Allerdings ist der breite Markterfolg der elektronischen Signatur bislang ausgeblieben [RoRo06. Mit der eCard-Strategie der Bundesregierung [Kowa07] und dem eCard-API-Framework [TR0312] war die Hoffnung verbunden, dass die starke Authentisierung und qualifizierte elektronische Signatur eines Tages leicht in beliebigen Anwendungen genutzt werden können. Doch auch dieser Ansatz scheint bislang nicht von Erfolg gekrönt. Vielmehr muss die Tatsache dass die AusweisApp des Bundes [BSI13b] bislang nur die eID-Funktionalität des neuen Personalausweises, aber weder andere Chipkarten (z.B. elektronische Gesundheitskarte, Heilberufsausweis, Signatur- und Bankkarten oder Ausweiskarten an-

P. Schartner · P. Trommler (Hrsg.) · D•A•CH Security 2013 · syssec (2013) 10-18.

derer EU-Mitgliedsstaaten) noch die erwarteten Signaturfunktionen unterstützt, als äußerst ernüchternd bewertet werden.

Vor dem Hintergrund dieser wenig befriedigenden Situation haben sich im Open eCard Team akademische und industrielle Experten zusammengefunden, um eine quelloffene und plattformunabhängige Implementierung des eCard-API-Frameworks [TR0312] bereitzustellen, durch die beliebige Anwendungen für Zwecke der Authentisierung und Signatur leicht auf beliebige Chipkarten zugreifen können.

In einer ersten Projektphase wurde dieses Rahmenwerk für die Realisierung einer leichtgewichtigen, vertrauenswürdigen und gleichsam gut bedienbaren Alternative zur AusweisApp des Bundes – der so genannten „Open eCard App“ (siehe auch [HPS+12] und [HHB+13]) – genutzt. Die Open eCard App wurde in Java entwickelt und kann als eigenständige Applikation oder als Applet im Browser ausgeführt werden. Die Quellen der Open eCard App sind unter http://source.openecard.org veröffentlicht. Anders als bei der AusweisApp des Bundes werden beliebige Chipkarten unterstützt, die durch CardInfo-Dateien gemäß ISO/IEC 24727 [ISO24727] beschrieben sind. Mit den kürzlich bereitgestellten Browsererweiterungen für Chrome, Opera, Safari und Firefox können die unter http://openecard.org/dienste aufgeführten Dienste genutzt werden. Außerdem existiert eine Android-basierte Version (http://openecard.org/android), die mit einem geeigneten NFC-fähigen Smartphone für die mobile Authentisierung mit dem neuen Personalausweis eingesetzt werden kann.

In einem zweiten Schritt und vor dem Hintergrund der während der Implementierung gesammelten Erfahrungen sowie den zunehmend klarer ersichtlichen Anforderungen wurde das ursprüngliche Design der Open eCard App [HPS+12] um einen Plugin-basierten Erweiterungsmechanismus ergänzt, durch den die grundlegende Open eCard Plattform um zusätzliche Funktionen ergänzt werden kann. Durch diesen Erweiterungsmechanismus können insbesondere entsprechende Signaturmodule ergänzt werden.

Der vorliegende Beitrag ist folgendermaßen gegliedert: In Abschnitt 2 werden die wichtigsten für die elektronische Signatur relevanten Aspekte des eCard-API-Frameworks [TR0312] grob zusammengefasst. Abschnitt 3 stellt das modulare und erweiterbare Design der Open eCard App vor und Abschnitt 4 beschreibt die Struktur eines entsprechenden Plugins für die Erstellung von CAdES- [ETSI101733], XAdES- [ETSI101903] und PAdES-Signaturen [ETSI102778]. Abschnitt 5 diskutiert beispielhafte Anwendungsfälle und Abschnitt 6 enthält eine kompakte Zusammenfassung und liefert einen Ausblick auf zukünftige Entwicklungen.

2 Signaturerzeugung mit dem eCard-API-Framework

Wie in [HFD+09] erläutert, wirken beim Erzeugen von elektronischen Signaturen typischerweise verschiedene Funktionen in den unterschiedlichen Schichten des eCard-API-Frameworks zusammen.

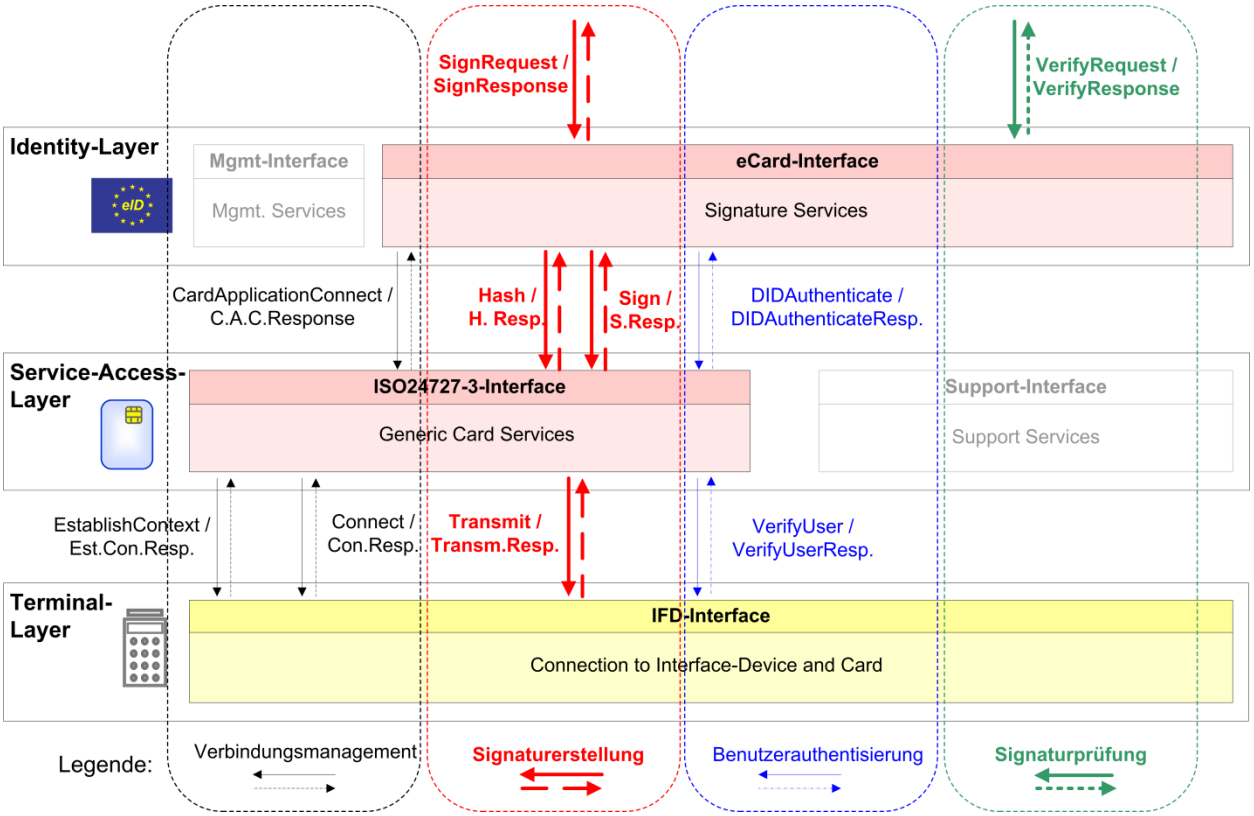

Abb. 1: Signatur-relevante Funktionen des eCard-API-Frameworks (aus [HFD+09])

Wie in Abbildung 2 angedeutet, löst hierbei ein `SignRequest` am eCard-Interface des Identity Layers (siehe [TR0312] (Teil 2) bzw. [OASIS07]) den Aufruf der Funktionen `Hash` und `Sign` aus. Für diese Aufrufe am Service Access Layer (SAL) (siehe [TR0312] (Teil 4) bzw. [ISO24727] (Teil 3) wird eine logische Verbindung zur Chipkartenapplikation benötigt, die mittels `CardApplicationConnect` aufgebaut wird.

Im Rahmen der Initialisierung des Frameworks wird für die Kommunikation zwischen dem SAL und dem Interface Device (IFD) – Layer (siehe [TR0312] (Teil 6) bzw. [ISO24727] (Teil 4) mittels `EstablishContext` ein Kommunikationskontext aufgebaut, in dem schließlich mit `Connect` technische Verbindungen zu den erfassten Chipkarten aufgebaut werden. Über eine solche technische Verbindung werden die im SAL gebildeten APDUs mit der `Transmit`-Funktion an den IFD–Layer gereicht und dort schließlich über das Chipkartenterminal und ein geeignetes Transportprotokoll zu einer kontaktbehafteten oder kontaktlosen Chipkarte übertragen.

In der Regel ist vor dem Zugriff auf den privaten Signaturschlüssel eine Benutzerauthentisierung erforderlich. Deshalb wird vor dem Aufruf der SAL-Funktion `Sign` die Funktion `DIDAuthenticate` aufgerufen, die wiederum die Funktion `VerifyUser` im IFD-Layer nutzt, um die Benutzerauthentisierung durch Eingabe einer PIN durchzuführen.

3 Design der Open eCard App

In diesem Abschnitt wird das erweiterbare Design der Open eCard App vorgestellt.

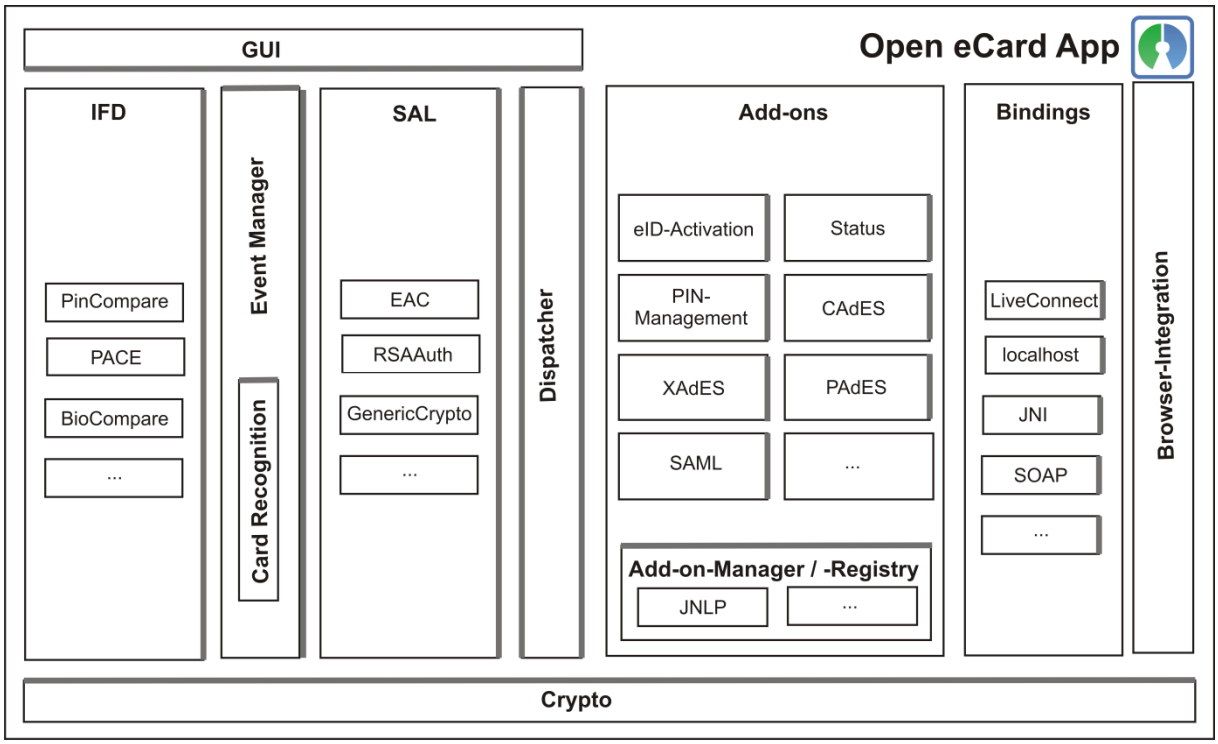

Abb. 2: Erweiterbare Architektur der Open eCard App

3.1 Die Open eCard Plattform

Die wesentlichen Module der grundlegenden Open eCard Plattform sind im Folgenden näher erläutert:

- **Interface Device (IFD):** Diese Komponente implementiert das IFD-Interface, das in TR-03112-6 [TR0312] und ISO/IEC 24727-4 [ISO24727] spezifiziert ist. Weiterhin enthält es zusätzliche Schnittstellen für Passwort-basierte Protokolle (z.B. PACE) und stellt eine einheitliche Schnittstelle für die Kommunikation mit unterschiedlichen Kartenlesern und Smartcards zur Verfügung.
- **Event Manager:** Der Event Manager überwacht eintreffende Events (z.B. das Hinzufügen oder Entfernen von Lesegeräten oder Karten) und führt die Typerkennung von neu hinzugefügten Karten durch. Da für die Erkennung des Chipkartentyps die in CEN 15480 [CEN15480] bzw. ISO/IEC 24727 [ISO24727] spezifizierten CardInfo-Dateien genutzt werden, kann die Open eCard App sehr leicht zusätzliche Chipkarten unterstützen.
- **Service Access Layer (SAL):** Dieses Modul implementiert den Service Access Layer wie er in TR-03112-4 [TR0312] und ISO/IEC 24727-3 [ISO24727] spezifiziert ist. Die Erweiterbarkeit dieser Komponente fußt auf zwei Konzepten. Einerseits ermöglicht das in Abschnitt 3.2 beschriebene „Add-on-Framework“ das Hinzufügen neuer Authentisierungsprotokolle, ohne andere Teile der Implementierung ändern zu müssen. Auf der anderen Seite können Erweiterungen kartenagnostisch implementiert werden, indem diese auf Informationen in CardInfo-Dateien zugreifen, die den Erweiterungen zur Verfügung

stehen. Somit kann allein durch das Hinzufügen einer CardInfo-Datei eine weitere Chipkarte unterstützt werden.

- **Crypto:** Das Crypto-Modul vereinheitlicht den Zugriff auf kryptographische Funktionen für andere Module. Durch die Nutzung der BouncyCastle Crypto Bibliothek [TLBC13] ist es möglich, die Open eCard App leicht auf Plattformen ohne vollständige Implementierung der Java Cryptographic Architecture (JCA) [Orac13a] zu portieren.
- **Graphische Benutzerschnittstelle (GUI):** Die GUI wird über eine abstrakte Schnittstelle angesprochen und ist jederzeit leicht gegen andere Implementierungen austauschbar. Dies ermöglicht Plattform-spezifische Implementierungen ohne eine Änderung anderer Module.
- **Dispatcher:** Der Dispatcher stellt eine zentrale Komponente dar, die alle ein- und ausgehenden Nachrichten an die entsprechenden Module weiterleitet. Diese Zentralisierung ermöglicht eine deutliche Reduzierung der Gesamtkomplexität. Zugleich ist es aber auch möglich zusätzliche Logik, die den Datenfluss steuert, einzubringen. Dadurch sind u.a. Filter für externe Nachrichten realisierbar.
- **Bindings:** Diese Komponente enthält verschiedene Bindings, die von den jeweiligen Schnittstellen der unten aufgeführten Plugins genutzt werden können. Hierbei kann das LiveConnect-Binding [Java], das localhost-Binding im Stile von [TR0312] (Teil 7, Abschnitt 3.2.1), das JNI-Binding [Orac13b] oder beispielsweise das SOAP-Binding [DDG+00] genutzt werden. Eine Übersicht, welche Plugins mit welchen Bindings genutzt werden können findet sich in [WHP+13].
- **Browser-Integration:** Damit die Open eCard App nicht nur in Verbindung mit fest installierten Fachanwendungen, sondern auch in Browser-gestützten Webanwendungen genutzt werden kann, werden verschiedene Möglichkeiten zur Browserintegration unterstützt. Dies umfasst die Bereitstellung der Open eCard App als Applet mit dem LiveConnect-Binding [Java], das localhost-basierte Binding, das durch die Verwendung von CORS [W3C12] eine äquivalente Funktionalität entfalten kann und längerfristig die engere Integration der Open eCard App in populäre Browser über die von diesen unterstützten kryptografischen Schnittstellen (z.B. PKCS#11 [RSAL09] oder CSP [MS13]).

3.2 Das Open eCard „Add-on-Framework"

Ein wesentlicher Aspekt der aktuellen Architektur der Open eCard App (siehe [WHP+13] und Abbildung 2) im Vergleich zur ursprünglichen Konzeption [HPS+12] ist der Erweiterungsmechanismus über Add-ons, die vom Add-on-Manager und der Add-on-Registry verwaltet werden. Hierbei wird – angelehnt an die HTML-Spezifikation [BFL+13] – bei den Add-ons zwischen Plug-ins, die über eine programmatische Schnittstelle angesprochen werden, und sonstigen Erweiterungen (Extensions) unterschieden. Über die Add-on-Registry kann bestimmt werden, welches Add-on für einen bestimmten Vorgang benötigt wird. Außerdem kümmert sich die Add-on-Registry bei Bedarf um die Beschaffung der entsprechenden Erweiterung. Beispielsweise kann eine solche Registry auf Basis von JNLP [Herr11] realisiert werden. In dieser Variante, die sowohl bei der Applet-basierten als auch bei der per Webstart ausgelieferten Open eCard App genutzt wird, ist die Registrierung der Erweiterungsmodule inhärent vorhanden. Die eigentlichen Add-ons werden über den in JNLP vorhandenen Mechanismus bedarfsgerecht nachgeladen und sodann für den schnellen Zugriff lokal vorgehalten. Längerfristig wären alternative Registrierungsmechanismen im Stile eines App Stores denkbar, die bei einer fest installierten Open eCard App zum Tragen kommen könnten.

Nach dem Laden des Erweiterungsmoduls durch die Add-on-Registry übernimmt der Add-on-Manager die Verwaltung der Add-on-Instanzen und kümmert sich um die Einhaltung von Sicherheitsrichtlinien durch einen Sandbox-Mechanismus.
Jedes Plug-in stellt in einer so genannten Manifest-Datei, deren Struktur in [WHP+13] näher beschrieben ist, eine Beschreibung der von ihm angebotenen Schnittstelle bereit, die mehrere Operationen beinhalten kann. Durch diese abstrakte Schnittstelle ist es möglich, das Binding, also die Schnittstelle zur Außenwelt, auszutauschen oder sogar mehrere Bindings zur Verfügung zu stellen.

3.3 Existierende und zukünftige Add-ons

Durch den oben erläuterten Add-on-Mechanismus kann die Open eCard Plattform um beliebige Funktionalität ergänzt werden, wobei derzeit insbesondere die folgenden Plugins vorgesehen sind:

- *eID-Activation* – enthält die Funktionalität für die eID-Aktivierung über localhost gemäß [TR0312] (Teil 7, Abschnitt 3.2.1) bzw. über serialisierte `object`-Elemente zur Unterstützung der bis 2014 dauernden Migrationsphase.
- *Status* – liefert Informationen über die aktuell verfügbaren Kartenterminals und die erfassten Karten.
- *PIN-Management* – erlaubt die Änderung und Entsperrung von PINs und das Setzen der Signatur-PIN im Zuge des Nachladens eines qualifizierten Zertifikates auf den nPA.
- *CAdES* – ermöglicht das Erstellen von Signaturen gemäß [ETSI101733].
- *XAdES* – ermöglicht das Erstellen von Signaturen gemäß [ETSI101903].
- *PAdES* – ermöglicht das Erstellen von Signaturen gemäß [ETSI102778].
- *SAML* – bietet schließlich Unterstützung für erweiterte SAML-Profile [OASIS11], und Bindings [OASIS10], die zu einem effizienteren und besser geschützten Authentisierungsablauf führen.

4 Grobe Architektur der AdES-Plug-ins

Die grobe Architektur für die Realisierung der Signaturfunktionalität ist in Abbildung 3 dargestellt. Hierbei existiert ein so genannter Request-Handler, der einen über ein Binding transportierten `SignRequest`-Aufruf gemäß [OASIS07] oder alternativ entsprechende Daten aus einem HTML-Formular (siehe [BFL+13], Abschnitt 4.11) entgegennimmt und sodann die Signaturerzeugung in der Kernkomponente (eSign-Core) initiiert. Diese Kernkomponente realisiert in Verbindung mit dem SAL die grundlegende und vom spezifischen AdES-Format unabhängige Funktionalität, wie z.B. das Lesen des Signaturzertifikates von der zu verwendenden Chipkarte (mit `CardApplicationPath`, `CardApplicationConnect`, `DIDList`, `DIDGet`, `DataSetSelect`, `DSIRead` und ggf. `DIDAuthenticate` für die Benutzerauthentisierung) und die spätere Erzeugung der Signatur (mit `Sign`). Die Durchführung entsprechender Transformationen bei der Erstellung einer XML-basierten Signatur, die Erzeugung des für die Signaturerzeugung vorgesehenen Hashwertes sowie die Komplettierung der erzeugten Signatur erfolgt jedoch nicht im generischen Kernmodul, sondern im Format-spezifischen Teilmodul für XAdES [ETSI101903], CAdES [ETSI101733] oder PAdES [ETSI102778].

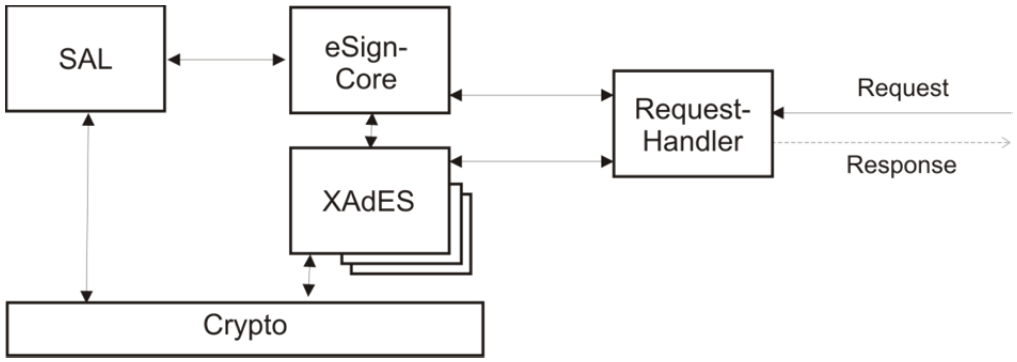

Abb. 3: Grobe Architektur der AdES-Plug-ins

5 Anwendungsbeispiel „Formularsignatur“

Die oben beschriebene Signaturfunktionalität der Open eCard App kann in unterschiedlichen Branchen und in vielfältigen Anwendungsfällen genutzt werden. Beispielhaft sollen im Folgenden die Abläufe bei der Erstellung einer Signatur für einen formulargebunden Geschäftsprozess in einer Webanwendung skizziert werden.

In diesem Fall können die Formulardaten entweder in der Webanwendung entsprechend aufbereitet und als OASIS-DSS-konformer `SignRequest` über ein geeignetes Binding (vgl. Abschnitt 3.1) an das eSign-Plug-in geschickt werden oder die Webanwendung schickt lediglich die Formulardaten wie in Abschnitt 4.11.22 von [BFL+13] beschrieben an die lokal operierende Open eCard App, die in diesem Fall die Aufbereitung der Formulardaten übernimmt und sodann die Signaturerzeugung wie oben erläutert initiiert und am Ende die auf diesem Weg erzeugte Signatur an die aufrufende Webanwendung zurückliefert.

6 Zusammenfassung und Ausblick

Der vorliegende Beitrag stellt das erweiterbare Design der Open eCard App vor und beleuchtet insbesondere wie auf dieser Basis die Funktionalität für die Erzeugung von elektronischen Signaturen in standardisierten Formaten wie XAdES [ETSI101903], CAdES [ETSI101733] oder PAdES [ETSI102778] umgesetzt werden kann. Hierbei können die auf OASIS-DSS [OASIS07] basierenden Schnittstellen aus Teil 2 des eCard-API-Frameworks [TR0312] oder alternativ HTML-basierte Mechanismen für die Signatur von Webformularen genutzt werden. Die hier beschriebene Signaturfunktionalität soll als Open Source bereitgestellt und einer möglichst breiten Nutzung in unterschiedlichen Anwendungsbereichen zugeführt werden.

Literatur

[BFL+13] R. Berjon, S. Faulkner, T. Leithead, E. Doyle Navara, E. O'Connor, S. Pfeiffer und I. Hickson: HTML 5.1 Nightly – A vocabulary and associated APIs for HTML and XHTML, Editor's Draft 2 July 2013. http://www.w3.org/html/wg/drafts/html/master/, 2013.

[BSI13a] Bundesamt für Sicherheit in der Informationstechnik (BSI): Ersetzendes Scannen, Technische Richtlinie, BSI-TR-03138, Version 1.0, 2013. http://resiscan.de

[BSI13b] Bundesamt für Sicherheit in der Informationstechnik (BSI): Offizielles Portal für die AusweisApp des Bundes, http://ausweisapp.bund.de, 2013.

[CEN15480] European Committee for Standardization (CEN): Identification card systems - European Citizen Card, Part 1 – 4, CEN/TS 15480, 2008.

[DDG+00] B. Don, E. David, K. Gopal, L. Andrew und M. Noah: Simple Object Access Protocol (SOAP) 1.1, 2000.

[ETSI101733] ETSI: CMS Advanced Electronic Signatures (CAdES), ETSI TS 101 733, Version 1.8.1, 2009.

[ETSI101903] ETSI: Technical Specification XML Advanced Electronic Signatures (XAdES), ETSI TS 101 903, Version 1.4.1, 2009.

[ETSI102778] ETSI: PDF Advanced Electronic Signature Profiles (PAdES), ETSI TS 102 778, Part 1-5, 2009.

[Herr11] A. Herrick: JSR 56: Java Network Launching Protocol and API, Maintainance Release 6, 2011.

[HFD+09] D. Hühnlein, S. Fischer-Dieskau, U. Gnaida, U. Korte, P. Rehäußer und W. Zimmer: „Langfristig beweiskräftige Signaturen mit dem eCard-API-Framework," in DACH Security 2009, 2009.

[HHB+13] M. Horsch, D. Hühnlein, C. Breitenstrom, T. Wieland, A. Wiesmaier, B. Biallowons, D. Petrautzki, S. Potzernheim, J. Schmölz, A. Wesner und T. Wich: Die Open eCard App für mehr Transparenz, Vertrauen und Benutzerfreundlichkeit beim elektronischen Identitätsnachweis, BSI-Kongress, Secumedia, 2013.

[HPS+12] D. Hühnlein, D. Petrautzki, J. Schmölz, T. Wich, M. Horsch, T. Wieland, J. Eichholz, A. Wiesmaier, J. Braun, F. Feldmann, S. Potzernheim, J. Schwenk, K. C., A. Kühne und H. Veit: On the design and implementation of the Open eCard App, GI SICHERHEIT 2012 Sicherheit – Schutz und Zuverlässigkeit, März 2012.

[HüKo06] D. Hühnlein und U. Korte: Grundlagen der elektronischen Signatur, Secumedia, 2006.

[ISO24727] ISO/IEC: Identification cards – Integrated circuit card programming interfaces, ISO/IEC 24727, Part 1 – 5.

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

[Kowa07] B. Kowalski: „Die eCard-Strategie der Bundesregierung im Überblick," in BIOSIG 2007: Biometrics and Electronic Signatures, 2007.

[MS13] Microsoft: „Cryptographic Service Providers". http://msdn.microsoft.com/en-us/library/windows/desktop/aa380245%28v=vs.85%29.aspx.

[OASIS07] OASIS: Digital Signature Service Core Protocols, Elements, and Bindings, Version 1.0, OASIS Standard, http://docs.oasis-open.org/dss/v1.0/oasis-dss-core-spec-v1.0-os.pdf.

[OASIS10] OASIS: SAML V2.0 Holder-of-Key Web Browser SSO Profile Version 1.0, http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-holder-of-key-browser-sso.pdf, 2010.

[OASIS11] OASIS: SAML V2.0 Enhanced Client or Proxy Profile Version 2.0, https://www.oasis-open.org/committees/download.php/41209/sstc-saml-ecp-v2.0-wd02.pdf, 2011.

[Orac13a] Oracle: Java Cryptography Architecture (JCA) Reference Guide, http://download.oracle.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html .

[Orac13b] Oracle: Java Native Interface. http://docs.oracle.com/javase/6/docs/technotes/guides/jni

[RoRo06] H. Rossnagel, H. Roßnagel: On Diffusion and Confusion – Why Electronic Signa-tures Have Failed, TrustBus 2006, SS. 71-80, 2006.

[RSAL09] RSA Laboratories: PKCS #11 Base Functionality v2.30: Cryptoki, 2009. http://www.cryptsoft.com/pkcs11doc/STANDARD/pkcs-11v2-30b-d5.doc

[TLBC13] The Legion of the Bouncy Castle: „Bouncy Castle Crypto API,“. http://www.bouncycastle.org/java.html

[TR0312] Bundesamt für Sicherheit in der Informationstechnik (BSI): eCard-API-Framework, Technical Guideline TR-03112, Part 1 – 7, Version 1.1.2, https://www.bsi.bund.de/ContentBSI/Publikationen/TechnischeRichtlinien/tr03112/index_htm.html.

[W3C12] W3C: Cross-Origin Resource Sharing, W3C Working Draft 3, http://www.w3.org/TR/cors, 2012.

[WHP+13] T. Wich, M. Horsch, D. Petrautzki, J. Schmölz, D. Hühnlein, T. Wieland und S. Potzernheim: An extensible client platform for eID, signatures and more, erscheint, 2013.