Die Open eCard App – Open Source für Authentisierung, Signatur und mehr

Detlef Hühnlein

Durch die eCard-Strategie der deutschen Bundesregierung[1] sollten die starke Authentisierung und die elektronische Signatur mit beliebigen Chipkarten und Anwendungen genutzt werden können. Während die vom Bund bereitgestellte AusweisApp[2] bislang leider nur den neuen Personalausweis unterstützt und deshalb diese Erwartungen nicht annähernd erfüllen konnte, haben sich im Open-eCard-Projekt[3] industrielle und akademische Experten zusammengeschlossen, um mit der Open eCard App[4,5,6,7] eine quelloffene und plattformunabhängige Implementierung des internationalen ISO/IEC 24727-Standards[8,9,10,11] bereitzustellen. Der vorliegende Beitrag stellt das erweiterbare Design der Open eCard App vor und zeigt, wie auf dieser Basis Funktionen für die starke Authentisierung und die elektronische Signatur realisiert werden können.

Dr. Detlef Hühnlein
Geschäftsführender Gesellschafter
ecsec GmbH
detlef.huehnlein@ecsec.de

Einleitung

Durch die elektronische Abwicklung von Geschäftsprozessen lassen sich Kosten senken sowie Fehlerquoten und Prozesslaufzeiten reduzieren. Dazu werden im Wesentlichen papiergebundene Schriftstücke zunehmend digitalisiert[12] oder durch elektronische Dokumente ersetzt und Prozesse automatisiert. Aus Sicherheitsgründen oder aufgrund von Formerfordernissen wird hierfür oftmals eine starke Authentisierung mit einem geeigneten Verfahren oder der Einsatz der elektronischen Signatur gefordert. Allerdings ist der breite Markterfolg der elektronischen Signatur bislang ausgeblieben[13]. Mit der eCard-Strategie der Bundesregierung[1] und dem eCard-API-Framework[11] war die Hoffnung verbunden, dass die starke Authentisierung und die qualifizierte elektronische Signatur eines Tages leicht in beliebigen Anwendungen genutzt werden könnten. Doch auch dieser Ansatz scheint bislang nicht von Erfolg gekrönt. Vielmehr muss die Tatsache, dass die AusweisApp des Bundes[2] bislang nur die eID-Funktionalität des neuen deutschen Personalausweises, aber weder andere Chipkarten (z. B. elektronische Gesundheitskarte, Heilberufsausweis, Signatur- und Bankkarten, Ausweiskarten anderer EU-Mitgliedsstaaten oder die Schweizer SuisseID) 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 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 sogenannten Open eCard App – genutzt. Die Open eCard App wurde in Java entwickelt und kann als eigenständige Applikation oder als Applet im Browser ausgeführt werden. Anders als bei der AusweisApp des Bundes werden beliebige Chipkarten unterstützt, die durch CardInfo-Dateien gemäss ISO/IEC 24727 beschrieben sind. Ausserdem existiert eine Android-basierte Version, 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[4] um einen Plug-in-basierten Erweiterungsmechanismus[6] ergänzt, durch den die grundlegende Open-eCard-Plattform um zusätzliche Funktionen ergänzt werden kann. Diese erweiterbare Architektur soll im Folgenden vorgestellt werden.

Die Architektur der Open eCard App

Das erweiterbare Design der Open eCard App ist in Abbildung 1 dargestellt.

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[11] und ISO/IEC 24727-4[8] spezifiziert ist. Weiterhin enthält es zusätzliche Schnittstellen für passwortbasierte 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[10] bzw. ISO/IEC 24727[8] spezifizierten CardInfo-Dateien genutzt werden, kann die Open eCard App sehr leicht zusätzliche Chipkarten unterstützen.

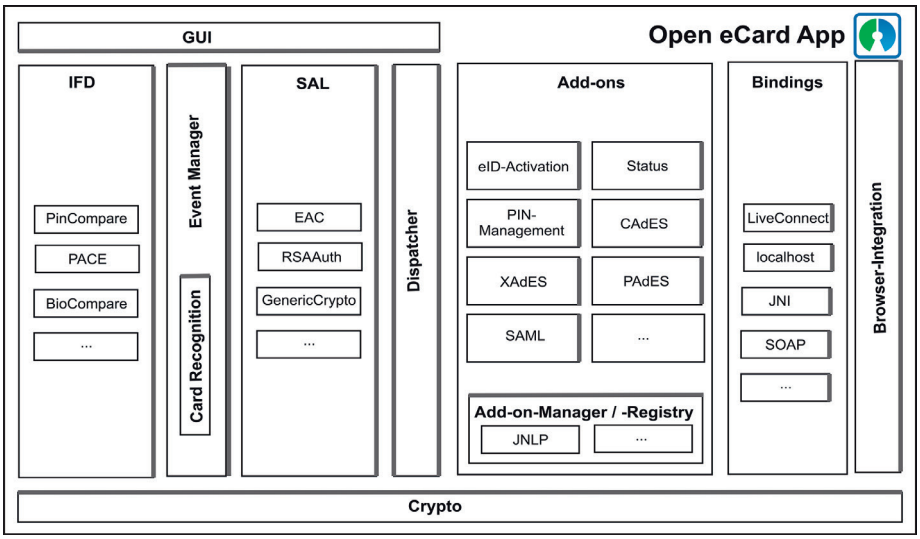

Abbildung 1: Erweiterbare Architektur der Open eCard App

Service Access Layer (SAL)

Dieses Modul implementiert den Service Access Layer, wie er in TR-03112-4[11] und ISO/IEC 24727-3[8] spezifiziert ist. Die Erweiterbarkeit dieser Komponente fusst auf zwei Konzepten: Einerseits ermöglicht das «Add-on-Framework»[6] 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 gemäss ISO/IEC 24727-3[8] 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 kryptografische Funktionen für andere Module. Durch die Nutzung der Bouncy-Castle-Crypto-Bibliothek[14] ist es möglich, die Open eCard App leicht auf Plattformen ohne vollständige Implementierung der Java Cryptographic Architecture (JCA)[15] zu portieren.

Grafische Benutzerschnittstelle (GUI)

Die GUI wird über eine abstrakte Schnittstelle angesprochen und ist jederzeit leicht gegen andere Implementierungen austauschbar. Dies ermöglicht plattformspezifische 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 unter anderem Filter für externe Nachrichten realisierbar.

Add-on-Framework

Ein wesentlicher Aspekt der aktuellen Architektur der Open eCard App[6] (siehe Abbildung 1) im Vergleich zur ursprünglichen Konzeption[4] 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[16] – 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 sie sich bei Bedarf um die Beschaffung der entsprechenden Erweiterung. Eine solche Registry kann beispielsweise auf Basis von JNLP[17] 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 sogenannten Manifestdatei[6] 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.

Bindings

Diese Komponente enthält verschiedene Bindings, die von den jeweiligen Schnittstellen der unten aufgeführten Plug-ins genutzt werden können. Hierbei kann das LiveConnect-Binding[18], das localhost-Binding im Stile von TR-03112[11] (Teil 7, Abschnitt 3.2.1), das JNI-Binding[19] oder beispielsweise das SOAP-Binding[20] genutzt werden.

Browserintegration

Damit die Open eCard App nicht nur in Verbindung mit fest installierten Fachanwendungen, sondern auch in browsergestützten Webanwendungen genutzt werden kann, werden verschiedene Möglichkeiten zur Browserintegration unterstützt. Dies umfasst die Bereitstellung der Open eCard App als Applet mit dem LiveConnect-Binding[21], das Localhost-basierte Binding, das durch die Verwendung von CORS[22] 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 oder CSP).

Authentisierung, Signatur und mehr

Durch die Erweiterungsmöglichkeiten über Protokolle und CardInfo-Dateien im SAL können leicht beliebige Chipkarten und Authentisierungsprotokolle unterstützt werden. Durch entsprechende Signatur-Plug-ins[23] für {X,C,P}AdES[24] können zukünftig auch elektronische Signaturen mit der Open eCard App erstellt werden. Darüber hinaus können durch den Erweiterungsmechanismus[6] aber auch leicht völlig andersartige Chipkartenanwendungen erstellt werden. Beispielsweise zeigen Kuhlisch et al.[25], wie mit der elektronischen Gesundheitskarte und der Open eCard App auf eine elektronische Patientenakte gemäss § 291a SGB V zugegriffen werden kann.

1 B. Kowalski: Die eCard-Strategie der Bundesregierung im Überblick. In: BIOSIG 2007: Biometrics and Electronic Signatures (2007).
2 Bundesamt für Sicherheit in der Informationstechnik (BSI): Offizielles Portal für die AusweisApp des Bundes, http://ausweisapp.bund.de (2013).
3 http://openecard.org.
4 D. Hühnlein, D. Petrautzki, J. Schmölz, T. Wich, M. Horsch, T. Wieland, J. Eichholz, A. Wiesmaier, J. Braun, F. Feldmann, S. Potzernheim, J. Schwenk, C. Kahlo, A. Kühne und H. Veit: On the Design and Implementation of the Open eCard App. In: GI SICHERHEIT 2012: Sicherheit – Schutz und Zuverlässigkeit, März 2012.
5 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.
6 T. Wich, M. Horsch, D. Petrautzki, J. Schmölz, D. Hühnlein, T. Wieland und S. Potzernheim: An extensible client platform for eID, signatures and more. In: Proceedings of Open Identity Summit, LNI 223 (2013). S. 55–69.
7 Siehe auch http://source.openecard.org.
8 ISO/IEC, Identification Cards – Integrated Circuit Card Programming Interfaces, ISO/IEC 24727, Part 1–5.
9 D. Hühnlein und M. Bach: How to Use ISO/IEC 24727-3 with Arbitrary Smart Cards. In: TrustBus 2007 (2007).
10 European Committee for Standardization (CEN), Identification Card Systems – European Citizen Card, Part 1–4, CEN/TS 15480 (2008).
11 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.
12 Bundesamt für Sicherheit in der Informationstechnik (BSI), Ersetzendes Scannen, Technische Richtlinie, BSI-TR-03138, Version 1.0, 2013.
13 H. Roßnagel: On Diffusion and Confusion – Why Electronic Signatures Have Failed, TrustBus 2006, SS. 71–80, 2006.
14 The Legion of the Bouncy Castle: Bouncy Castle Crypto API (Online). http://www.bouncycastle.org/java.html.
15 Oracle, Java Cryptography Architecture (JCA) Reference Guide. http://download.oracle.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html.
16 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).
17 A. Herrick, JSR 56: Java Network Launching Protocol and API, Maintainance Release 6, 2011.
18 Java.net, LiveConnect Support in the New Java™ Plug-in Technology, http://jdk6.java.net/plugin2/liveconnect/.
19 Oracle, Java Native Interface, http://docs.oracle.com/javase/6/docs/technotes/guides/jni/.
20 B. Don, E. David, K. Gopal, L. Andrew und M. Noah: Simple Object Access Protocol (SOAP) 1.1, 2000.
21 Java.net, LiveConnect Support in the New Java™ Plug-in Technology, http://jdk6.java.net/plugin2/liveconnect/.
22 W3C, Cross-Origin Resource Sharing, W3C Working Draft 3, http://www.w3.org/TR/cors (2012).
23 D. Hühnlein, J. Schmölz, T. Wich und A. Kühne: Elektronische Signaturen mit der Open eCard App, DACH Security 2013.
24 http://www.e-signatures-standards.eu/
25 R. Kuhlisch, D. Petrautzki, J. Schmölz, B. Kraufmann, F. Thiemer, T. Wich, D. Hühnlein und T. Wieland: An Open eCard Plug-in for Accessing the German National Personal Health Record, Proceedings of Open Identiy Summit 2013, LNI 223 (2013), S. 82–93.