Moritz Horsch, David Derler, Christof Rath, Hans-Martin Haase, Tobias Wich

# Open Source für europäische Signaturen

## Vertrauenswürdige Basis für die elektronische Signatur

**Moritz Horsch**

ist wissenschaftlicher Mitarbeiter an der Technischen Universität Darmstadt und forscht im Bereich der sicheren Authentisierung.

E-Mail: horsch@cdc.informatik.tu-darmstadt.de

**David Derler**

ist wissenschaftlicher Mitarbeiter am Institut für Angewandte Informationsverarbeitung und Kommunikationstechnologie (IAIK) an der Technischen Universität Graz. Forschungsgebiete: Digitale Signaturen, Privacy Enhancing Technologies.
E-Mail: david.derler@iaik.tugraz.at

**Christof Rath**

ist seit vier Jahren wissenschaftlicher Mitarbeiter an der TechnischenUniversität Graz und beschäftigt sich mit sicherer Datenverarbeitung und eGovernment Themen.
Email: christof.rath@iaik.tugraz.at

**Hans-Martin Haase**

ist Berater und Softwareentwickler bei der ecsec GmbH und beschäftigt sich mit den Themen Smart Cards und Identity Management.

E-Mail: hans-martin.haase@ecsec.de

**Tobias Wich**

beschäftigt sich seit mehreren Jahren bei der ecsec GmbH mit dem Thema Authentisierung für das Cloud Computing und promoviert in diesem Bereich an der Ruhr-Universität Bochum.
E-Mail: tobias.wich@ecsec.de

Obwohl die elektronische Signatur oft als Schlüsselkomponente für die vertrauenswürdige Abwicklung von elektronischen Geschäftsprozessen gilt, wird sie bisher in der Praxis kaum genutzt. Durch die kommende eIDAS-Verordnung der Europäischen Kommission werden positive Impulse und eine Belebung des Europäischen Signaturmarktes erwartet. Für die Steigerung von Vertrauen, Transparenz und Akzeptanz auf Nutzer- und Serviceseite, bedarf es unter anderem einer flexiblen, interoperablen, benutzerfreundlichen und als Open Source verfügbaren Signaturanwendung.

### 1 Einleitung

Mit der Europäischen Signaturrichtlinie 1999/93/EU [1] wurde ein Rahmenwerk für die elektronische Signatur in Europa geschaffen, das durch individuelle Signaturgesetze der Mitgliedsstaaten weiter detailliert und in nationales Recht umgesetzt wurde. Da die Signaturgesetze in Europa leider in Details voneinander abweichen, ist der grenzüberschreitende Einsatz der elektronischen Signatur mit Schwierigkeiten verbunden. Um diese Hindernisse für den Europäischen Binnenmarkt zu überwinden wurde von der Europäischen Kommission mit COM(2012) 238 [2] im Juni 2012 ein Entwurf für eine EU-Verordnung vorgelegt, die unmittelbar gelten wird und neben den Vertrauensdiensten für die elektronische Signatur auch das elektronische Siegel und die elektronische Identifizierung regeln soll. Auf Basis dieses Entwurfs und der teilweise kritischen Kommentare [3] [4] sind inzwischen Änderungsvorschläge des Europäischen Rates und des Europäischen Parlaments in den Verordnungsentwurf [5] eingeflossen und der Abschluss des Verfahrens wird in Kürze erwartet. Aspekte der Interoperabilität sollen durch weitere Rechtsakte und Europäische Normen geregelt werden, die derzeit unter dem Standardisierungsmandat M/460 [6] von CEN und ETSI erarbeitet werden. Damit für die Nutzung der elektronischen Signatur in Europa zukünftig flexible, interoperable und benutzerfreundliche Signaturanwendungen bereitstehen, soll im Rahmen des von der EU geförderten FutureID Projektes [7] ein geeigneter Signatur-Client entwickelt und als Open Source bereitgestellt werden. Der vorliegende Beitrag erläutert die Vorüberlegungen, beschreibt die Architektur der geplanten Signaturanwendungen

und geht schließlich auf mögliche Anwendungsszenarien und Sicherheitsaspekte ein.

### 1.1 Existierende Ansätze

Für den geplanten Signatur-Client empfiehlt sich ein Blick auf existierende Basiskomponenten und Signaturbibliotheken sowie darauf aufbauender Client- und Serverkomponenten.

**Kryptographische Basiskomponenten**
Zu den bekanntesten Open Source Bibliotheken für Verschlüsselung und Signatur zählt OpenSSL [8]. Die unter einer BSD-artigen Lizenz verfügbare Bibliothek unterstützt die gängigen Signaturverfahren. Eine ähnliche Funktionalität bietet die in C++ entwickelte CryptoPP [9] oder die BouncyCastle Bibliothek [10] für Java und C#.

**Signaturbibliotheken**
Für Zwecke der elektronischen Signatur existieren beispielsweise die im OpenXAdES Projekt entstandene DigiDoc Bibliothek [11], die unter der LGPL Lizenz verfügbar ist, und die kryptographischen Basis- und Signaturbibliotheken [12] der Stiftung Secure Information and Communication Technologies (SIC) des Institutes für Angewandte Informationsverarbeitung und Kommunikationstechnologie (IAIK) der TU Graz, die unter einer kommerziellen Lizenz vertrieben werden.

**eID- und Signaturkomponenten**
Aufbauend auf den oben genannten Basiskomponenten wurden bereits verschiedene eID- und Signaturanwendungen entwickelt, die möglicherweise als Grundlage für einen Open Source Signatur-Client genutzt werden könnten. Hier sind insbesondere die unter der European Union Public License (EUPL) verfügbare österreichische MOCCA Plattform [13], die belgischen eID- und Signaturkomponenten [14], [15], die portugiesische Cartão de Cidadão Middleware [16], die OpenSC Middleware [17], die deutsche AusweisApp des Bundes [18], das PersoApp Projekt [19], der Sirius Signaturserver [20], das von der Europäischen Union unterstützte Digital Signature Service Projekt [21] und nicht zuletzt die erweiterbare Open eCard App [22] zu nennen.

## 2 Client

Das Ziel des FutureID Projektes [7] ist es, eine zugleich privatsphären-freundliche und, sowohl von Nutzern als auch von Service Providern, einfach zu verwendende Infrastruktur für elektronische Authentisierung und Signatur bereitzustellen. Eine zentrale Rolle spiel dabei eine Client Applikation (im folgenden FutureID Client genannt), welche als Schnittstelle zwischen dem Nutzer und den von Dienstanbietern angebotenen Anwendungen fungiert und die starke Authentisierung mit beliebigen Authentisierungstoken (u.a. den in Europa eingesetzten Ausweis- und Signaturkarten) sowie die Erstellung von „fortgeschrittenen elektronischen Signaturen“ gemäß EU-Signaturrichtlinie [1] bzw. zukünftig auch der Europäischen eIDAS-Verordnung [6], [5] unterstützt. Hierfür wurden die in Abschnitt 1.1 erwähnten Ansätze und Komponenten näher analysiert und auf dieser Basis eine auf den relevanten Standards, wie z.B. ISO/IEC 24727 [23], JCA/JCE [24] und OASIS-DSS [25], aufbauende Architektur entwickelt, die im weiteren Verlauf des FutureID Projektes auf Basis der erweiterbaren Open eCard Plattform [22] und der IAIK Signaturbibliotheken [12] umgesetzt wird und wieder in das Open eCard Projekt zurückfließen soll.

Im Folgenden bietet dieser Abschnitt einen Überblick über den FutureID Client sowie das Add-on Framework und das darauf aufbauende Signatur Plug-in.

### 2.1 Wesentliche Komponenten

Der FutureID Client zeichnet sich durch einen modularen Aufbau aus, was eine leichte Erweiterbarkeit sicherstellen soll. Die für die Signaturfunktionalität erforderlichen Komponenten sind im Folgenden erläutert. Eine Übersicht über den gesamten Client findet sich beispielsweise in [22] und [26].

- Interface Device (IFD)
  Der IFD ermöglicht den Zugriff auf unterschiedliche Kartenleser und Smartcards über eine einheitliche Schnittstelle gemäß TR-03112-6 [27] und ISO/IEC 24727-4 [23].
- Service Access Layer (SAL)
  Diese Komponente implementiert den in TR-03112-3 [27] sowie ISO/IEC 24727-3 [23] spezifizierten SAL. Neben dem Zugriff auf Smartcards und der Verwaltung von Kartenapplikationen ist der SAL insbesondere für die Durchführung unterschiedlicher Authentisierungsprotokolle zuständig. Des Weiteren werden Protokolle für das Erstellen von Signaturen auf Smartcards vom SAL zur Verfügung gestellt.
- Add-on Framework
  Das Add-on Framework stellt die einfache Erweiterbarkeit des Clients sicher und wird in Abschnitt 2.2 näher diskutiert. Eine ausführliche Beschreibung ist in [22] zu finden.
- Bindings
  Dieses Modul enthält unterschiedliche Bindings, die von den Plug-ins genutzt werden können. Hervorzuheben ist insbesondere das localhost Binding [27] (Teil 7, Abschnitt 3.2.1).

### 2.2 Add-on Framework

Das Add-on Framework erlaubt eine einfache Erweiterbarkeit des Clients über nachladbare Softwaremodule, sogenannte Add-ons. Es wird dabei zwischen den vier Typen *Application Extensions*, *Application Plug-ins* und *Protocol Plug-ins* für IFD und SAL unterschieden, die zur Erweiterung verschiedener Komponenten des Client verwendet werden können. Über Application Plug-Ins werden zusätzliche Dienste bereitgestellt, die über die verschiedenen Bindings (wie zum Beispiel HTTP POST) angesprochen werden können. Ein Beispiel dafür ist das Signatur Plug-in welches in Abschnitt 2.3 vorgestellt wird. Mittels Protocol Plug-ins können IFD bzw. SAL Protokolle zur Verfügung gestellt werden, um zum Beispiel neue eID-Karten bzw. zusätzliche Funktionalitäten dieser Karten anzusprechen. Application Extensions erlauben es unter anderem Erweiterungen mit graphischer Benutzeroberfläche anzubieten.

Das Add-on Framework sieht vor, dass Add-ons von unterschiedlichen Quellen – sogenannten *AddonRegistries* – bezogen werden können. Beispiele dafür wären ein denkbarer *FutureID AppStore* oder kommerzielle AppStores, wie man sie von diversen Smartphone Plattformen kennt.

Die zentrale Komponente im Add-on Framework ist der *AddonManager* der für das Laden von Add-ons zuständig ist und

dabei beliebig viele Add-on Quellen unterstützt. Eine Frage, die dabei in den Mittelpunkt rückt: Welche Rechte haben solche Add-ons, beziehungsweise dürfen solche Add-ons haben, wenn sie auf der Client-Plattform ausgeführt werden? Leider kann auf diese Frage keine universelle Antwort gegeben werden, weil verschiedene Add-on Quellen mit großer Wahrscheinlichkeit mit verschiedenen Vertrauensniveaus assoziiert werden. So könnten Add-ons, die direkt mit dem Client ausgeliefert werden, beispielsweise die gleichen Berechtigungen eingeräumt werden wie dem Client selbst; bei einem Add-on aus einem AppStore sollten diese Rechte hingegen eingeschränkt werden. Um dies zu erreichen ist das Add-on Framework so gestaltet, dass jede AddonRegistry auch die Definition bzw. Implementierung einer für diese AddonRegistry geltenden Richtlinie (Policy) vorschreibt.

### 2.3 Signatur Plug-in

Das auf dem Add-on Framework aufbauende Signatur Plug-in fungiert als Signaturserver und ermöglicht Signaturen mit Hilfe von standardisierten OASIS DSS [25] Anfragen erstellen bzw. verifizieren zu lassen. Vorteil dieser Architektur ist es, dass die Signaturerstellung vollkommen von der aufrufenden Applikation entkoppelt ist, sowohl hinsichtlich des gewünschten Signaturformates als auch hinsichtlich des verwendeten Signaturerstellungsgerätes, wie z.B. einer Smartcard. So wird die aufrufende Applikation nicht, beziehungsweise nur indirekt, durch mögliche Änderungen an Signaturstandards oder Signaturgesetzen beeinflusst und Änderungen können an zentraler Stelle im Client (Signatur Plug-in) eingepflegt werden. Ausgehend von den DSS Anfragen entscheidet das Signatur Plug-in über das zu verwendende Signaturformat und ggf. die zu verwendende Signaturerstellungseinheit und gibt die entsprechende Signatur bzw. das entsprechende Verifizierungsergebnis zurück. Zusätzlich wird die Benutzeroberfläche durch eine Application Extension erweitert, um eine Signaturerstellung, bzw. -verifikation direkt aus dem Client auszulösen.

Aus dem Gesichtspunkt der Signaturverifizierung agiert der Client standardmäßig wie ein Proxy, welcher die Verifizierungsanfragen an ein Validierungsservice weiterleitet. Der Vorteil davon ist, dass gewisse Entscheidungen, wie zum Beispiel der Vertrauensstatus von Zertifikaten, nicht vom Client getroffen werden müssen, was eine bessere Usability für unerfahrene Nutzer verspricht. Nichtsdestotrotz bietet die Client Architektur die Möglichkeit einen eigenen Service zu betreiben und alle relevanten Entscheidungen selbst zu treffen.

Sowohl bei der Signaturverifizierung als auch bei der Signaturerstellung werden die maßgeblichen Europäischen Standards für XML- [28], PDF- [29] und CMS- [30] basierte fortgeschrittene elektronische Signaturen unterstützt, wobei für die Erstellung der Signaturen in den zuvor genannten Signaturformaten am Client externe Libraries des IAIK [12] verwendet werden. Wichtige Voraussetzung, um externe Libraries in der bestehenden Architektur von SAL und IFD verwenden zu können, ist es, dass diese Libraries für die Berechnung des Signaturwertes intern auf die Java Cryptography Architektur (JCA) zurückgreifen. Dies ermöglicht es, verschiedene Java Cryptographic Provider (JCP) für verschiedene Arten der Signaturerstellung zur Verfügung zu stellen. Ein Beispiel für einen solchen JCP ist ein Provider für das Erstellen von Signaturen auf Smartcards über SAL bzw. IFD Protokolle. Weitere Signaturszenarien, welche unter anderem über einen solchen Provider realisiert werden könnten, werden im folgenden Abschnitt diskutiert.

## 3 Anwendungsszenarien

Die elektronische Signatur ermöglicht eine Vielzahl von Anwendungen. Wir betrachten die lokale und serverseitige Signaturerstellung sowie eine On-demand Signaturzertifikatserstellung. Die folgenden Abschnitte erläutern jeweils die einzelnen Szenarien und beschreiben die Vor- und Nachteile sowie die mögliche technische Umsetzung im FutureID Client.

### 3.1 Lokale Signatur

Der erste Anwendungsfall betrachtet das klassische Szenario einer lokalen Signaturerzeugung. Die Auswahl des Signaturschlüssels, die Berechnung des Hash-Wertes, die Durchführung des Signaturverfahrens und das Erzeugen der Signatur im jeweiligen Format erfolgt dabei durch den Client.

Eine lokale Signaturerzeugung kann auf drei verschiedene Arten durchgeführt werden. Alle drei Varianten übergeben dabei eine DSS [25] Anfrage an das Signatur Plug-in des Clients:

- In der ersten Konstellation erfolgt das Auslösen der Signaturfunktion durch eine externe Anwendung. Die Anwendung selektiert (ggf. mit Hilfe des Benutzers) das zu signierende Dokument und erstellt eine passende DSS Anfrage. Die Anfrage wird mittels HTTP POST [31] an das localhost Binding des Clients (http://127.0.0.1:24727/eSign) gesendet. Das Binding leitet die Anfrage dann an das Signatur Plug-in weiter, welches wiederrum die weiteren Schritte der Signaturerstellung durchführt.
- Als zweite Möglichkeit können externe Services auch mittels der Sign-Funktion vom SAL eine DSS Anfrage an den Client stellen, die dann wiederrum an das Plug-in weitergereicht wird.
- Als dritte Möglichkeit kann eine eigene Signaturfunktionalität des Clients bequem durch eine Application Extension realisiert werden, die einen Auswahldialog für das zu signierende Dokument bereitstellt und eine DSS Anfrage erzeugt.

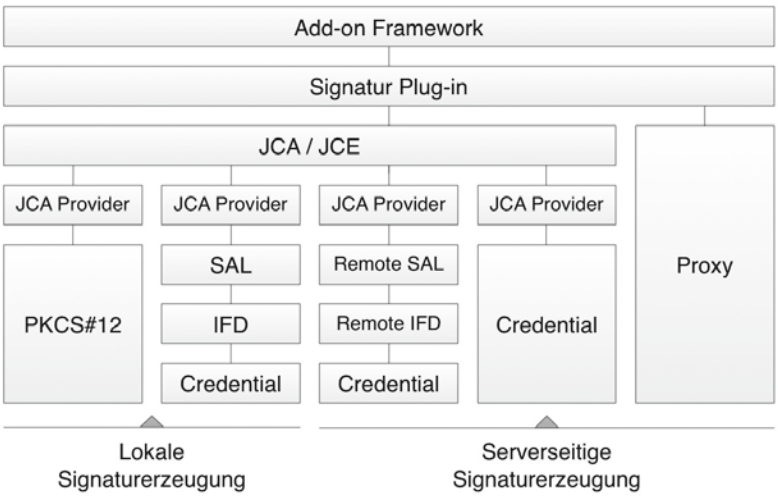
Das Plug-in übernimmt die Auswahl des Signaturzertifikats bzw. Signaturschlüssels, die Signaturerstellung und das Generieren des entsprechenden Signaturformates. Die kryptografischen Funktionen werden über den in Abschnitt 2.2 beschriebenen Provider durchgeführt, der beispielsweise eine Smartcard über den SAL bzw. IFD anspricht oder einen Schlüssel aus einem PKCS#12 KeyStore [32] verwendet (siehe Abbildung 1).

Die lokale Signatur zeichnet sich insbesondere dadurch aus, dass der Nutzer und der Client die alleinige und vollständige Kontrolle über die Signaturerzeugung haben. Auf der anderen Seite fordert dies eine gewisse Funktionalität und Ressourcen vom Client und der Client Plattform.

### 3.2 Server Signatur

Eine Signaturerzeugung benötigt Funktionalitäten wie das Berechnen eines Hash-Wertes, die mathematische Berechnung einer digitalen Signatur und das Erzeugen des jeweiligen Signaturformates. Insbesondere im mobilen Bereich sind ggf. nicht alle Geräte in der Lage diese Anforderungen zu erfüllen. So müssten beispielsweise elliptische Kurzen unterstützt werden, um ECDSA [33] Signaturen zu erstellen, oder XML Objekte erstellt werden, wenn XAdES [28] Signaturen verwendet werden. Insbesondere für ressourcen-beschränkte Geräte ist dies unter Umständen eine Herausforderung.

**Abbildung 1 | Signaturerstellung**




Ein weiteres Problem ist die sichere Speicherung der privaten Schüssel. Technologien wie *Secure Elements* (SE) oder *Trusted Execution Environments* (TEE) könnten hier einen sicheren Schlüsselspeicher bieten. Jedoch sind die Verbreitung und insbesondere die Nutzung dieser Technologien für Anwendungen beschränkt und oft nur durch Gerätehersteller nutzbar. Bei Server Signaturen befindet sich daher der private Schlüssel des Nutzers auf einem entfernten Rechner, der damit die Signaturerzeugung durchführt. Die Berechnung des Hash-Wertes und das Generieren des Signaturformates können, in Abhängigkeit des Funktionsumfangs des Gerätes und des Sicherheits- bzw. Vertrauensniveaus, durch den Server oder die Client Plattform des Nutzers geschehen. Erfolgt die Erzeugung des Hash-Wertes durch den Client ist für den Server der Inhalt des zu signierenden Dokuments nicht ersichtlich.

In Abhängigkeit der Funktionalität des Clients bieten sich hier zwei Lösungen an.

- Ist der Server ausschließlich für das Erzeugen der Signatur zuständig, d.h. das Erzeugen des Hash-Wertes und des Signaturformats erfolgt durch den Client, dann kann das Auslagern dieses Vorgangs über einen weiteren JCA Provider erfolgen. Anstatt den lokalen SAL bzw. IFD zu nutzen kann der Provider die Kommunikation mit dem Signaturserver kapseln. Denkbar wäre selbstverständlich auch ein externer SAL bzw. IFD (siehe Abbildung 1).
- Erfolgt der gesamte Signaturerstellungsprozess inklusive Hash-Wertberechnung und Formaterstellung ebenfalls durch den Server können DSS Anfragen durch das Plug-in oder durch das Binding an den externen Signaturserver weitergeleitet werden. Dieser führt dann die gesamte Signaturerstellung durch und übermittelt das Ergebnis zurück an den Client, der wiederrum das Ergebnis nur weiterreicht. Diese Konstellation ist in Abbildung 1 durch den Proxy dargestellt.

Ein wichtiges Kriterium für die Server Signatur ist eine transparente Durchführung. Dies bedeutet, dass für eine Anwendung nicht ersichtlich ist, dass die Signatur auf einem Server erstellt wurde.

Betrachten wir die Sicherheit einer serverseitigen Signatur ist insbesondere hervorzuheben, dass der Server durch den Besitz des privaten Schüssels in der Lage wäre, beliebige Dokumente im Namen des Nutzers zu signieren. Daher muss zwischen Nutzer und Server eine adäquate Vertrauensbeziehung existieren, was wiederum geeignete Maßnahmen zur sicheren gegenseitigen Authentisierung erfordert.

### 3.3 On-demand Signatur

Um die Verbreitung der elektronischen Signatur weiter voran zu treiben bieten sich sogenannte On-demand Signaturen an.

Fordert beispielsweise eine Anwendung eine Signaturerzeugung oder möchte der Nutzer ein Dokument signieren, verfügt aber über kein, oder kein geeignetes, Signaturzertifikat, kann er ein passendes Zertifikat in wenigen Schritten bei einem externen Zertifizierungsdienstleister beantragen. Dazu muss sich der Nutzer gegenüber dem Dienstleister mit einem geeigneten Protokoll authentisieren. Entscheidend ist dabei, dass der Prozess der Zertifikatsbeantragung in möglichst wenigen und einfachen Schritten erfolgt. Dies kann ggf. eine vorherige Registrierung des Nutzers bei dem Zertifizierungsdienstleister erfordern. Aus Gründen der Benutzerfreundlichkeit und des Datenschutzes wäre eine Beantragung mit wenigen Aktionen und über standardisierte Schnittstellen zu begrüßen, damit der Nutzer beispielswiese bei jeder Zertifikatbeantragung einen anderen Zertifizierungsdienstleister wählen kann.

In Abhängigkeit des Anwendungsfalls und insbesondere der zu signierenden Daten kann die Authentisierung unterschiedlich stärk ausgeprägt sein. Eine Authentisierung auf Basis von Benutzername und Passwort ist sicherlich für viele Anwendungsfälle ausreichend, während eine Authentisierung beispielsweise mit dem neuen Personalausweis die Ausstellung eines Zertifikates für eine qualifizierte elektronische Signatur ermöglichen würde.

On-demand Signaturen können vielseitig ausgeprägt sein. Die Gültigkeit des Signaturzertifikates kann beispielsweise auf wenige Stunden oder Tage begrenzt sein. Prinzipiell denkbar wäre auch, dass beispielsweise mit dem Merkle Signatur Verfahren [34] die Anzahl der möglichen Signaturen auf eine bestimmte Anzahl begrenzt wird. Ebenfalls könnte die Signatur auf einen Dienstanbieter beschränkt werden, so dass nur dieser in der Lage ist die Signatur zu verifizieren. Mit diesen vielseitigen Möglichkeiten lassen sich zahlreiche Anforderungen im Bereich des Datenschutzes sowie unterschiedliche Vertrauens- und Sicherheitsniveaus umsetzen.

Wie bei der Server Signatur ist es auch bei der On-demand Signatur essenziell, dass die Zertifikatserzeugung auf Abruf vollkommen transparent gegenüber der Anwendung abläuft. Das bedeutet, dass es für die Anwendung nicht ersichtlich ist ob der Benutzer nur ein passendes Signaturzertifikat auswählt oder ein komplett neues Zertifikat für diesen Anwendungsfall durch den externen Zertifizierungsdienstleister erstellt wird.

Das vorgestellte Signatur Plug-in erfüllt diese Anforderungen. Das Plug-in ist für das Laden des privaten Schlüssels und dem Setzen der Parameter für die Signaturerzeugung zuständig. Diese Funktionalität lässt sich bequem anpassen, erweitern oder austauschen.

Im einfachsten Fall wählt das Plug-in das Zertifikat bzw. den privaten Schüssel automatisch aus. Es könnten aber auch komple-

xe Auswahldialoge für den Benutzer bereitgestellt werden, oder wie im Falle einer On-demand Signatur ein Signaturzertifikat erst bei Bedarf beantragt werden. Das Plug-in implementiert dazu die passenden Benutzerinteraktionen wie beispielsweise die Auswahl des externen Signatur- oder Zertifizierungsdienstleisters und die Eingabe von erforderlichen Daten. Ist der Nutzer bereits bei einem Dienstleister registriert, können Authentisierungsdaten abgefragt werden. Das Plug-in erstellt dann einen Schlüssel und einen Zertifikatsantrag gemäß PKCS#10 [35] und schickt diesen über einen vertrauenswürdigen Kanal an einen geeigneten Zertifizierungsdienstleister. Sobald das Zertifikat empfangen wird, kann das Plug-in wie bei der klassischen lokalen Signaturerstellung fortfahren.

Weitere Vorteile der On-demand Signatur liegen im Bereich der Privatsphäre. Unter der Voraussetzung, dass die Beantragung des Signaturzertifikates so einfach wie möglich gestaltet wird, könnten Nutzer bequem kurzlebige Einmalsignaturzertifikate erhalten, die möglicherweise für jede Signatur von einem anderen Zertifizierungsdienstleister stammen.

## 4 Sicherheit

Das Vertrauen in die Anwendung ist insbesondere bei der elektronischen Signatur von größter Bedeutung. Dabei ist es beispielsweise essenziell, darauf vertrauen zu können, dass die Anwendung auch tatsächlich nur das vom Benutzer gewünschte Dokument signiert. Dies kann durch eine vertrauenswürdige Anzeigekomponente (Trusted Viewer) sichergestellt werden, die Gegenstand einer Zertifizierung sein kann.

Die Überprüfung dieses Sicherheitsversprechens durch den Benutzer bleibt aber eine Herausforderung, so dass es letztendlich auf Vertrauen in die Integrität der genutzten Plattform hinausläuft. Ein anderer, für das Vertrauen in eine Signaturapplikation wichtiger Punkt, ist der angemessene Umgang mit privatem Schlüsselmaterial.

Eine transparente Entwicklung, frei verfügbare Dokumentationen und Spezifikationen sowie quellenoffene Applikationen können dabei maßgeblich das Vertrauen in eine Anwendung stärken. Der vorgestellte FutureID Client bietet diese Transparenz und Möglichkeit zur Verifikation der Sicherheit und trägt damit einen entscheiden Beitrag zur Förderung der Akzeptanz der elektronischen Signatur bei.

## 5 Fazit

Die beschriebenen Szenarien einer lokalen, serverseitigen und On-demand Signatur ermöglichen ein breites Spektrum von Anwendungen. Jedes setzt dabei verschiede Akzente und ermöglicht den Einsatz der elektronischen Signatur unter den unterschiedlichsten Voraussetzungen. Dazu zählen Funktionsumfang und Ressourcen des Clients und der Client Plattform sowie das gewünschte Datenschutz-, Vertrauens- und Sicherheitsniveau. Zusätzlich bietet die Architektur und Modularität des FutureID Clients eine breite Unterstützung für verschiedene Smartcards zur Signaturerzeugung.

Die Quellenoffenheit des Clients stärkt zusätzlich das Vertrauen, die Transparenz und die Sicherheit von Signaturanwendungen und trägt damit maßgeblich zum Erfolg und zur Akzeptanz der elektronischen Signatur in Europa bei.

## Literatur


[1] Europäischen Union, Richtlinie 1999/93/EG des Europäischen Parlaments und des Rates vom 13. Dezember 1999 über gemeinschaftliche Rahmenbedingungen für elektronische Signaturen, 2000.

[2] Europäische Kommission, Verordnung des europäischen Parlaments und des Rates über die elektronische Identifizierung und Vertrauensdienste für elektronische Transaktionen im Binnenmarkt.

[3] Gisela Quiring-Kock, Entwurf EU-Verordnung über elektronische Identifizierung und Vertrauensdienste, DuD 2013, Heft 1, S. 20-24.

[4] Stephan Sädtler, "Identity management in cloud computing in conformity with European Union law? – Problems wand approaches pursuant to the proposal for a regulation by the European Commission on electronic identification and trust services for electronic transactions.," in *Proceedings of Open Identity Summit 2013, GI LNI 223*, Seiten 118-129, 2013.

[5] Council of the European Union, Proposal for a Regulation of the European Parliament and of the Council on electronic identification and trust services for electronic transactions in the internal market – Consolidated text, 2013.

[6] European Commission, Standardisation Mandate to the European Standardisation Organisations CEN, CENELEC and ETSI in the field of Information and Communication Technologies Applied to Electronic Signatures, 2009.

[7] FutureID, 2013.

[8] Eric A. Young and Tim J. Hudson. http://www.openssl.org.

[9] Wei Dai. http://www.cryptopp.com.

[10] Bouncy Castle. http://www.bouncycastle.org

[11] DigiDoc libraries. http://id.ee/index.php?id=30486

[12] Stiftung Secure Information and Communication Technologies. Secure Information and Communication Technologies. http://jce.iaik.tugraz.at/

[13] MOCCA: Modular Open Citizen Card Architecture Project.

[14] Cornelis F., eID-Applet Project.

[15] Cornelis F., eID Digital Signature Service Project.

[16] Middleware do Cartão de Cidadão. http://svn.gov.pt/projects/ccidadao

[17] M. Paljak, OpenSC Project – tools and libraries for smart card.

[18] Bundesamt für Sicherheit in der Informationstechnik (BSI), AusweisApp.

[19] PersoApp – Sichere und benutzerfreundliche Internet-Anwendungen. https://www.persoapp.de

[20] Kühne A. and Veit H., Sirius Sign Server Project.

[21] David Naramski. Open Signature Service. https://joinup.ec.europa.eu/software/sd-dss/description

[22] Tobias Wich, Dirk Petrautzki, Johannes Schmölz, Moritz Horsch, and Detlef Hühnlein, An extensible platform for eID, signatures and more, 2013.

[23] ISO/IEC, Identification cards – Integrated circuit card programming interfaces.

[24] Sun Inc., Java Cryptographic Architecture (JCA).

[25] OASIS Digital Signature Services TC, Digital Signature Services – DSS Core Protocols, Elements, and Bindings v1.0, 2007.

[26] Detlef Hühnlein, „Die Open eCard App – Open Source für Authentisierung, Signatur und mehr," *eGov-Präsenz*, vol. 01, no. http://ecsec.de/pub/2014_eGov.pdf, pp. 72-73, 2014.

[27] Bundesamt für Sicherheit in der Informationstechnik (BSI), eCard-API-Framework.

[28] European Telecommunication Standards Institute, Electronic Signatures and Infrastructures (ESI); XML Advanced Electronic Signatures (XAdES); ETSI TS 101 903, 2010.

[29] European Telecommunication Standards Institute, Electronic Signatures and Infrastructures (ESI); PDF Advanced Electronic Signatures (PAdES); ETSI TS 102 778, 2009.

[30] European Telecommunication Standards Institute, Electronic Signatures and Infrastructures (ESI); CMS Advanced Electronic Signatures (CAdES); ETSI TS 101 733, 2008.

[31] Fielding R. et al., Hypertext Transfer Protocol – HTTP/1.1, 1999.

[32] RSA Laboratories, PKCS #12 Personal Information Exchange Syntax, 2012.

[33] National Institute of Standards and Technology (NIST), Digital Signature Standard (DSS), FIPS PUB 186-4, 2013.

[34] R. Merkle, "A certified digital signature," in *Advances in Cryptology – CRYPTO '89*, 1989.

[35] Nystrom M. and Kaliski B., PKCS #10: Certification Request Syntax Specification.

[36] Richtlinie 1999/93/EG des Europäischen Parlaments und des Rates vom 13. Dezember 1999 über gemeinschaftliche Rahmenbedingungen für elektronische Signaturen, 2000.