Dokumentation

PDF-Verifikation vom Papierausdruck

Dokumentation

Version 0.1, 20.12.2016

Tobias Kellner – tobias.kellner@egiz.gv.at

Zusammenfassung: In diesem Projekt wurde ein Proof of Concept entwickelt, wie im Zuge einer digitalen Signaturaufbringung das signierte Dokument sicher abgelegt werden kann, um in der Signatur einen QR Code zu hinterlegen, mittels dem auf das abgelegte digitale Dokument zum Zweck der Signatur- und Dokumentenprüfung zugegriffen werden kann.

E-Government Innovationszentrum
Inffeldgasse 16/a, A-8010 Graz
Tel. +43 316 873 5514
Fax. +43 316 873 5520
E-Mail post@egiz.gv.at
www.egiz.gv.at

Das E-Government Innovationszentrum ist eine gemeinsame Einrichtung des Bundeskanzleramtes und der TU-Graz

BUNDESKANZLERAMT ÖSTERREICH
TU Graz
Graz University of Technology

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Überblick

Mit der Abschaffung der textuellen PDF-AS-Signatur gibt es keine einfache Möglichkeit mehr, wie ein digital signiertes Dokument von seinem Papierausdruck geprüft werden kann. Mit diesem Proof of Concept wird ein Weg aufgezeigt, wie diese Überprüfung vom Papierausdruck möglich wäre.

Der einfachste Möglichkeit dafür stellt ein QR-Code [QR] dar, der, in die Signatur als Bildmarke eingefügt, zu einer digitalen Version des ausgedruckten Dokuments führt. Um Vertraulichkeit zu gewährleisten muss dieses online abgelegte Dokument allerdings verschlüsselt werden.

1.2 Anforderungen

Die grundsätzlichen Anforderungen sind:

- Einfache Bedienung
- Vertraulichkeit

wobei Vertraulichkeit bedeutet, dass möglichst nur der Unterzeichner sowie der, der den QR-Code scannt Zugriff auf das signierte Dokument haben sollen, und nicht beispielsweise der Cloud-Provider der das Dokument speichert. Deshalb muss das Dokument verschlüsselt hinterlegt werden.

2 Konzept

2.1 Überblick

Zur Umsetzung wurde eine einfache Web-Anwendung gewählt, die PDF-AS für den Signaturvorgang verwendet. Der Benutzer, der ein Dokument signieren möchte, ruft die Webanwendung auf. Dabei wird im Browser des Signierenden ein geheimer symmetrischer Schlüssel generiert, mit dem das Dokument verschlüsselt werden soll. Die Web-Anwendung stellt eine ID zur Verfügung, mittels der später auf das hinterlegte Dokument zugegriffen werden soll.

Daraufhin wird der Signaturvorgang gestartet. Dabei wird auch gleich der Link auf das später zu hinterlegende Dokument übergeben, damit dieser von PDF-AS als QR-Code in die Signatur eingefügt werden kann. Nach erfolgreicher Signatur wird, wiederum rein im Browser des Signierenden, das signierte Dokument heruntergeladen, verschlüsselt, und mit der zuvor generierten ID an die Webanwendung übermittelt.

Das signierte Dokument enthält nun einen QR-Code, der auf eine online hinterlegte Version des Dokuments verweist. Wenn nun ein Benutzer diesen Code scannt, wird er wiederum auf die Webanwendung geleitet, die das unter der entsprechenden ID hinterlegte, verschlüsselte Dokument zur Verfügung stellt. Dieses Dokument wird nun im Browser der überprüfenden Benutzers entschlüsselt – mittels des geheimen Schlüssels, der im Fragment der URL im QR-Code hinterlegt wurde, und somit nicht zum Server-Teil der Webanwendung übermittelt wird. Abschließend wird das nun entschlüsselte Dokument angezeigt, und kann zur Überprüfung an PDF-AS übermittelt werden, um automatisch die Gültigkeit der Signatur anzuzeigen.

2.2 Sicherheit und Vertraulichkeit

2.2.1 Annahmen

Der PDF-AS-Anwendung muss in Punkto Vertraulichkeit und Sicherheit vollständig vertraut werden, da diese ohnehin das unverschlüsselte Dokument verarbeiten muss – sowohl bei der Signatur, als auch bei der Signaturprüfung bei Anzeige des Dokuments.

Der Cloud-Speicher, bei dem das Dokument hinterlegt wird, wird als „Honest but curious“ angenommen. Er liefert also das unveränderte zuvor hinterlegte Dokument. Wäre er „Malicious“, könnte er die erfolgreiche Entschlüsselung des Dokuments verhindern (Denial of Service), aber nicht den Inhalt ändern.

Schließlich wird angenommen, dass die Web-Anwendung in der derzeit vorliegenden Form unverändert zur Verfügung gestellt wird, um eine Speicherung oder Modifikation der Daten auszuschließen.

2.2.2 Analyse

Das Dokument wird zuerst (mittels TLS geschützt) von der Webanwendung an PDF-AS übermittelt. Des Weiteren wird dabei die URL, die im QR-Code hinterlegt werden soll, übermittelt, mittels der man später Zugriff auf das online gespeicherte Dokument hat. Hier muss also dem Server, der die PDF-AS-Anwendung zur Verfügung stellt, vertraut werden, dass er diese Daten vertraulich behandelt und diese nicht verändert oder gespeichert werden.

Danach wird das signierte Dokument im Browser des signierenden Benutzers mittels des zuvor generierten geheimen Schlüssels verschlüsselt. Das so verschlüsselte Dokument wird an den Cloud-Provider übertragen.

Dieser kann nun nicht mehr auf den Inhalt des Dokuments zugreifen. Wäre er „Malicious", könnte er zwar die Verschlüsselten Daten verändern, dies würde aber aufgrund der Verwendung von Message Authentication Codes bei der Entschlüsselung bemerkt werden, und stellte so lediglich ein Denial of Service dar.

Bei der Überprüfung des signierten Dokuments scannt der überprüfende Benutzer den in der Signatur abgebildeten QR-Code. Dabei wird das in der URL enthaltene Schlüsselmaterial an den Browser des Benutzers übertragen und dort temporär zur Entschlüsselung gespeichert. Vom Cloud-Provider wird das verschlüsselte Dokument bezogen, und daraufhin rein im Browser des überprüfenden Benutzers entschlüsselt und angezeigt.

Bei der anschließenden Signaturprüfung wird das Dokument an PDF-AS übertragen, dem dabei wiederum Punkto Vertraulichkeit vertraut werden muss.

3 Umsetzung

3.1 Überblick

Mit dem hier vorliegenden Proof of Concept wurde das oben dargelegte Konzept umgesetzt; es wurde dabei Wert auf eine möglichst einfache Demonstration des Konzepts gelegt.

Die Client-Seite der Webanwendung besteht aus JavaScript-Code. Zur Verschlüsselung wird die Web Cryptography API [Crypto] verwendet, zur temporären Speicherung von Daten im Browser HTML5 Local Storage [Storage].

Die Server-Seite wurde rudimentär in PHP umgesetzt und speichert lediglich hinterlegte verschlüsselte Dokumente bzw. stellt diese zur Verfügung. Dies ist lediglich ein Platzhalter zu Demonstrationszwecken für einen richtigen Cloud-Provider. Aufgrund fehlender Sicherheitsüberprüfungen könnte diese Anwendung zum Speichern beliebiger Daten missbraucht werden.

3.2 Ablauf

3.2.1 Initialer Aufruf der Webanwendung

- Der signierende Benutzer ruft die Webanwendung auf
- **URL**: https://demo.egiz.gv.at/pdf-print-verify/
- **Datei**: `index.php`

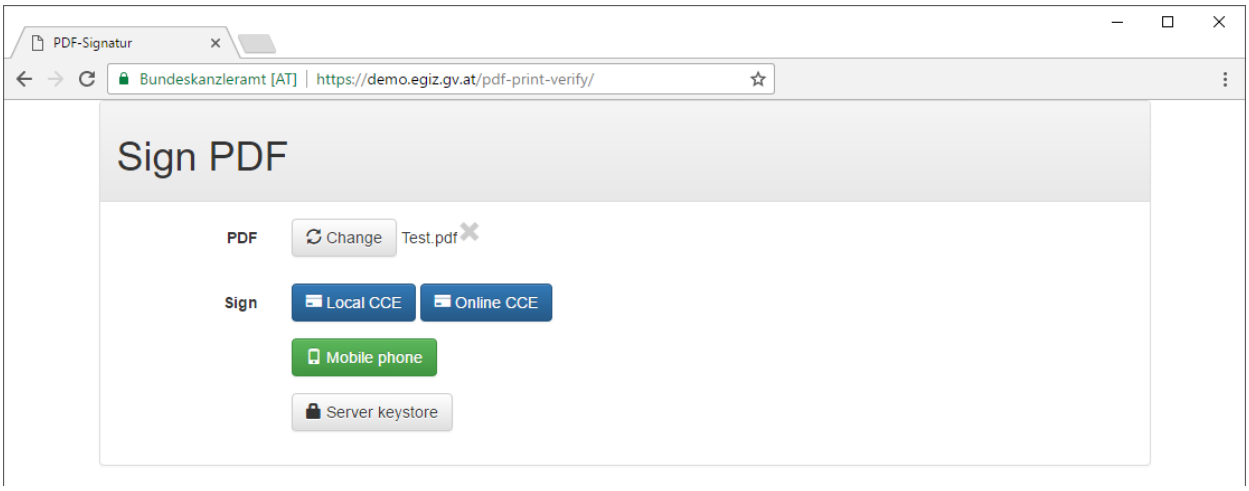

Figure 3.1 – Initialer Aufruf

Die Webanwendung generiert eine ID (`ID`), mittels derer später das Dokument später hinterlegt und verfügbar gemacht werden soll, und überträgt diese an den Browser des Benutzers. Dort wird zusätzlich das geheime symmetrische Schlüsselmaterial erzeugt und temporär gespeichert – im sessionStorage-Speicher. Das Schlüsselmaterial besteht dabei aus einem Initialisierungs-Vektor (`IV`) sowie einem symmetrischen Schlüssel (`Secret`). Außerdem wird das zu signierenden Dokument gehasht, und dieser Hash-Wert ebenfalls hinterlegt. Schließlich wird die URL erzeugt, die im QR-Code abgebildet wird, und unter dem später das Dokument abrufbar sein soll.

Diese sieht wie folgt aus:
https://demo.egiz.gv.at/pdf-print-verify/check?<ID>#<Secret>,<IV>

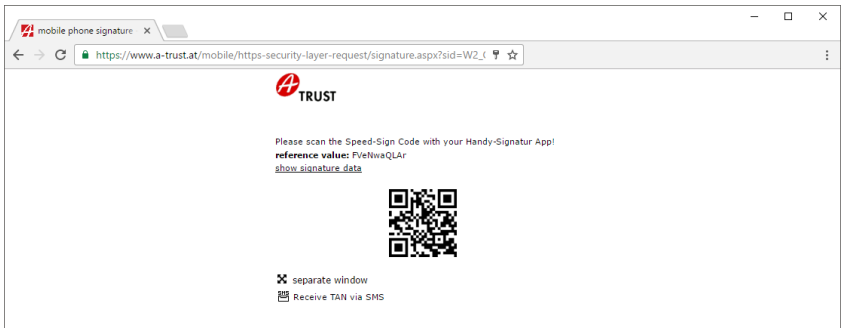

Figure 3.2 – Signaturvorgang

3.2.2 Signatur

- Der signierende Benutzer schickt das Dokument zur Signatur an die PDF-AS-Webanwendung
  - Parameter, die an PDF-AS übertragen werden:
    - qrcontent: <QR-Code-URL>
    - fileinput: <Zu signierendes Dokument>
- Dort wird der Signaturvorgang durchgeführt
- Anschließend wird der Callback der ursprünglichen Webanwendung aufgerufen
- **URL**: https://demo.egiz.gv.at/pdf-print-verify/result
- **Datei**: `result.php`

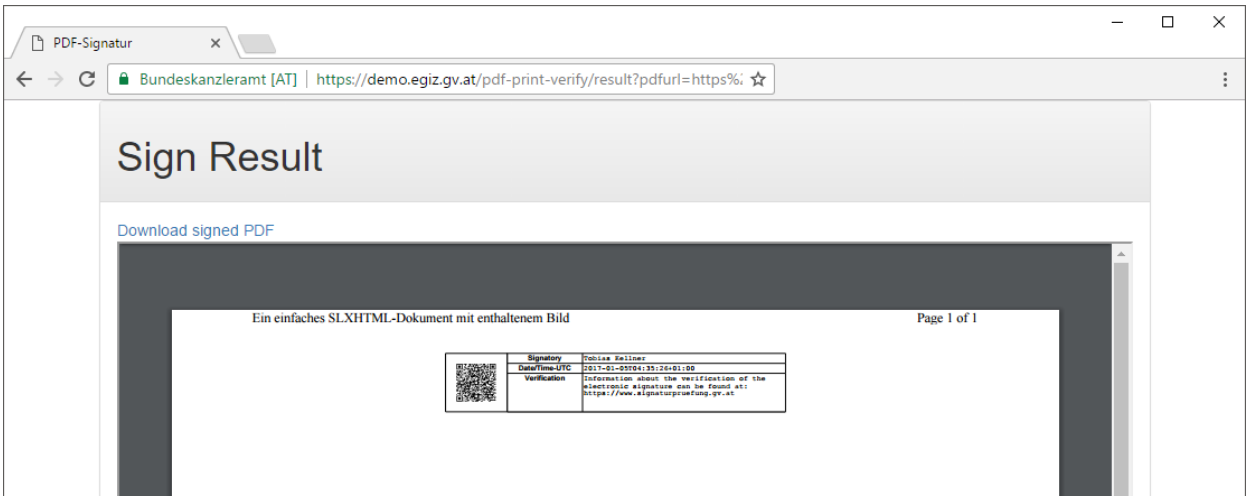

Figure 3.3 – Erfolgte Signatur

Das erfolgreich signierte Dokument wird nun aus dem Client-Teil der Webanwendung von PDF-AS in den Browser des Benutzers geladen. Dabei wird zur Überprüfung der Korrektheit der zuvor ermittelte Hash des Original-Dokuments an PDF-AS übermittelt.

Dieses Dokument wird nun dem Benutzer angezeigt, und gleichzeitig im Browser mit den zuvor erzeugten geheimen symmetrischen Schlüssel-Daten verschlüsselt. Das so verschlüsselte Dokument wird nun, zusammen mit der von der Webanwendung generierten ID, an den „Cloud-Speicher" übertragen, welcher in diesem Proof of Concept von der Serveranwendung `post.php` simuliert wird.

3.2.3 Überprüfung

- Der überprüfende Benutzer scannt den QR-Code im visuellen Signaturblock des zuvor signierten Dokuments
- Dabei wird er an die Webanwendung verwiesen
- **URL**: https://demo.egiz.gv.at/pdf-print-verify/check?<ID>#<Secret>,<IV>
- **Datei:** `check.php`

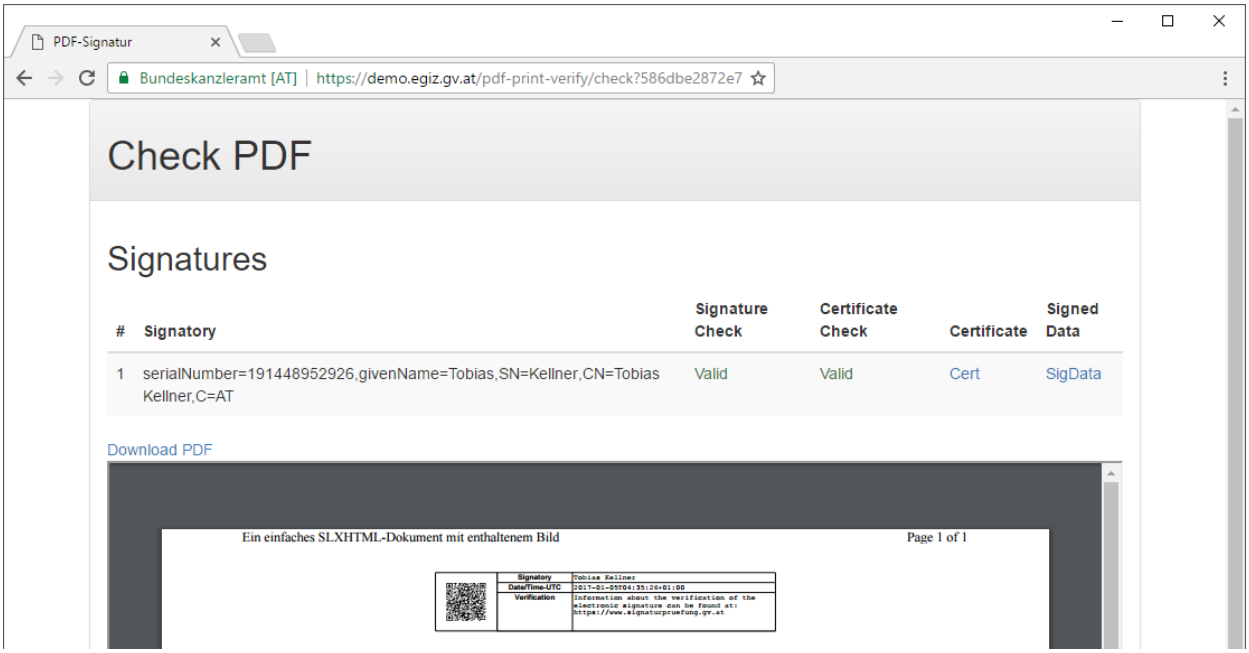

Figure 3.4 – Dokumentenüberprüfung

Im Browser des überprüfenden Benutzers werden nun die ID sowie die symmetrischen Schlüssel-Daten aus der URL geparst. Da sich die Schlüssel-Daten im Fragment der URL befinden, werden diese auch nicht an den Server-Teil übertragen, befinden sich also nur im Client-Browser. Nun wird das verschlüsselte Dokument vom „Cloud-Speicher" bezogen, welcher wiederum von der Serveranwendung `fetch.php` simuliert wird. Dieses verschlüsselte Dokument wird mit den zuvor ermittelten Schlüssel-Daten entschlüsselt und dem Benutzer angezeigt.

Abschließend wird das entschlüsselte Dokument an die PDF-AS-Webanwendung zur Überprüfung der Signaturen übermittelt, und das Ergebnis dem Benutzer angezeigt.

So kann der Benutzer überprüfen, ob der Inhalt des digital signierten Dokuments mit dem Ausdruck übereinstimmt, und ob die enthaltenen Signaturen gültig sind.

4 Anhang

Dokumentenhistorie

Version	Datum	Autor(en)	Anmerkung
0.1	20.12.2016	Tobias Kellner	Initialversion

Referenzen

[QR] https://en.wikipedia.org/wiki/QR_code
[Crypto] https://w3c.github.io/webcrypto/Overview.html
[Storage] http://www.w3schools.com/html/html5_webstorage.asp