HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL

FHO Fachhochschule Ostschweiz

PrivaSphere™

Long-Term Validation für PDF Signaturen

Bachelorarbeit

Abteilung Informatik
Hochschule für Technik Rapperswil

Herbstsemester 2017

Autor:	Alexis Suter
Betreuer:	Prof. Andreas Steffen
Projektpartner:	PrivaSphere
Experte:	Dr. oec. Ralf Hauser
Gegenleser:	Dr. Peter Sommerlad

Erstelldatum	Dienstag, 21. November 2017
Änderungsdatum	Samstag, 30. Dezember 2017
Druckdatum	Samstag, 30. Dezember 2017
Autoren	Alexis Suter (alexis.suter@gmail.com)

Management Summary

Ausgangslage

Für digitale Signaturen gibt es bereits gut etablierte Standards, welche langsam aber sicher ihren Einzug in die Schweiz erreichen. Der vom ETSI entwickelte LTV-Standard eignet sich hervorragend für den Ersatz der handschriftlichen Unterschrift. Die sog. Long Term Validation bezeugt, dass die Signatur mit einem gültigen Zertifikat versehen ist und ist mit einem zeitlichen Siegel versehen.

Da es für den LTV-Standard für PDF-Dokumente wenige Lösungen gibt, soll eine von Kopierschutz frei verfügbare Entwicklung erbracht werden. Als Basisbibliothek für PDF und Umsetzungsort steht Apache PDFBox basierend auf Java fest.
Nicht zu Letzt soll die LTV-Validierung auch nach dem Signieren ermöglicht werden, um diese Prozesse voneinander abzukoppeln.

Vorgehen / Technologien

Durch LTV werden einige Standards tangiert. Einer der wichtigen ist PDF, welcher viele sehr logische Definitionen in sich trägt. LTV wurde von ETSI entwickelt und erweitert den PDF-Standard. Damit ist klar definiert, wo und wie Signaturen und Validitätsdaten hingehören. Die Signaturen sind in CMS (Cryptographic Message Syntax) definiert und mit ASN.1 strukturiert.

Zeitliche Siegel entstehen durch sogenannte Trusted Timestamps (extern signierte Zeitstempel) um die Existenz einer Signatur an einem Zeitpunkt zu beweisen.

Für die Validierung müssen Dienste der Zertifikatausssteller angefragt werden um zu erfahren ob das eingesetzte Zertifikat gültig ist. Diese Antworten werden dann in das PDF-Dokument eingebunden. Die Dienste basieren entweder auf OCSP oder CRL.

Ergebnisse

Es wurden mehrere Teilschritte der LTV und damit alle Schritte in PDFBox implementiert.

- Es kann nachträglich ein signierter Zeitstempel in die Signatur eingebunden werden.
- Es gibt eine neue Zeitstempel-Signatur, welche der herkömmlichen PDF-Signatur ähnlich ist.
- Der grösste Teil bildet die Validierung der Zertifikate. Sämtliche gesammelte Daten werden in einer definierten Struktur in das PDF eingebunden.

Die Lösung wurde in Java mithilfe von PDFBox und BouncyCastle erstellt.

Aufgabenstellung

HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

Bachelorarbeit 2017

Long-Term Validation für PDF Signaturen

Studenten: Alexis Steiner

Betreuer: Prof. Dr. Andreas Steffen

Ausgabe: Montag, 18. September 2017

Abgabe: Freitag, 22. Dezember 2017

Einführung

A PDF signature may not be successfully verified unless its collateral validation components are preserved, e.g., certificates, CRLs, time stamp tokens, revocation lists, and OCSP responses. To facilitate long term signature validation (LTV), PDF supports the ability to collect validation information to verify a signature at a later time if it has been verified once as being valid.

Some of this information, i.e. certificates, CRLs and OCSP responses, when not already present in the signature, shall be stored in a Document Security Store (DSS). When storing this type of information and, when not already present in the signature, it shall be stored in a Document Time-Stamp (DTS) dictionary. This will provide the information needed to verify a signature as this was done when that signature was first verified.

If someone signs a PDF off-line, there should be a PDFBox routine that can possibly even be run on the command-line to amend a document with OCSP/CRL info for the signing certificate chain plus a verification time-stamp. The latter might even be interesting for an online signature that already has a timestamp but might be lacking other info.

Aufgabenstellung

- Einarbeiten in die relevanten LTV Standards und in die Apache PDFBox Software.
- Erweiterung der Apache PDFBox Funktionalität, so dass die Integration eines nachträglichen Timestamps inklusive zusätzlicher Signatur in ein signiertes PDF Dokument möglich ist.
- Erweiterung der Apache PDFBox Funktionalität, so dass die Integration von OCSP Responses in ein signiertes PDF Dokument möglich ist.
- Erfolgreiches Testen der Eweiterungen mit dem Acrobat Reader und dem Validator der Schweizerischen Bundesverwaltung.

Rapperswil, 18. September 2017

a. steffen

Prof. Dr. Andreas Steffen

INS -Institute for Networked Solutions 18.09.2017 Steffen

Inhalt

1 Einführung

1.1 Problemstellung und Vision

Das digitale Signieren bzw. Unterschreiben von Dokumenten ist ein sehr aktuelles Thema. Vor allem soll damit geholfen werden einiges an Papierarbeit auf den Computer zu transferieren. Die Möglichkeit ein Dokument digital zu unterschreiben gibt es schon länger. Das Vorgehen ist in etwa das gleiche wie die Signierung von Emails.

Der Schweizer Bund hat hierzu Regelungen für die digitale Signatur gestellt, die auf den detaillierten Spezifikationen des ETSI basieren. Der wichtigste Unterschied zur herkömmlichen Signatur ist, dass die Signatur zeitlich abgesichert und im Vornherein validiert werden muss. Im Groben ist das Ziel dieser Arbeit, eben diese Art von Signatur zu untersuchen und zu implementieren. Hauptgrund hierfür ist, dass keine quelloffene Lösung zur Verfügung steht.

Als zusätzliches Problem stellt das nachträgliche Validieren der Dokumente dar. Dies geht mit dem Interesse einher, eine möglichst sichere Signierung zu gestalten.

1.2 Ziele

Die LTV – Long Term Validation einer Signatur ist der Standard, der für die Validierung und Absicherung bereitsteht und vom ETSI definiert wurde. Hierfür sollen diverse Standards und Vorgehensweisen analysiert bzw. verstanden und wo nötig eingesetzt werden.

1.3 Rahmenbedingungen

Die Umsetzung der LTV soll auf Apache PDFBox basieren, einer Open Source Java Bibliothek für die Arbeit an PDF-Dokumenten. Vorteil ist, dass die Apache Lizenz offen und pragmatisch ist. Dadurch kann der umgesetzte Code frei eingesetzt und kopiert werden. Zudem erlaubt die Apache Gemeinschaft auch eine Beteiligung von Externen (Contributors).

Wichtig, ist dass die LTV mitsamt Zeitstempel nachträglich erstellt werden kann. Dies um den Prozess der Signatur von der Validierung abzukoppeln.

1.4 Vergleich mit vorhandenen Lösungen

Es gibt bisher wenige Lösungen zur LTV. Der Adobe Acrobat Reader ist eines der verbreitetsten Programme, welches diese implementiert. Jedoch ist diese nicht quelloffen und kann somit nicht in eigene Programme implementiert werden.
Weiter gibt es eine Implementierung durch iText. Diese ist steht aber unter Copyleft, was beim Einsatz der Library zu einem Copyright-Problem führen kann.

Somit ist PDFBox ideal, da es wirklich Quelloffen ohne Copyleft ist.[1]

2 Umsetzungskonzept

Dieses Kapitel beschreibt die benötigten Standards mit deren Zusammenhänge.

2.1 Gesetze Schweiz

Die Schweiz hat im Jahr 2003 das Bundesgesetz über die elektronische Signatur, ZertES[2] veröffentlicht. Dieses legt den Grundstein für die digitale Unterschrift. Eine Sammlung der Gesetze, Vorschriften und Beschreibungen befindet sich auf der BAKOM-Seite[3]. Diese Gesetze schreiben zwar einiges zur Handhabung der Zertifikate und Zertifizierungsdienste vor, aber nicht zur Signatur selber. Hierfür werden die internationalen Standards anerkannt, die dem aktuellen Stand der Technik entsprechen sollen.

Das BAKOM übernimmt viele Normen um mit Europa zu harmonisieren[4]. Eines der europäischen Normen-Institute ist das ETSI, welches auch Normen zur digitalen Signatur herausgegeben hat. Wichtig sind vor allem Spezifikationen zur Signatur selber und zu deren Umgang mit PDF-Dokumenten.

2.1.1 PAdES

***P**DF **Ad**vanced **E**lectronic **S**ignatures* wurde durch ETSI definiert und regelt die Anpassungen und Einschränkungen basierend auf dem PDF-Standard um Signaturen Standard-konform einzubringen. Ähnliche Erweiterungen von Standards gibt es auch für CMS und XML. Es wird aber nicht weiter darauf eingegangen, da sie keinen Einfluss auf die Arbeit haben.

PAdES wurde in verschiedene Profile unterteilt. Wichtig sind die Profile PAdES-Basic und PAdES-LTV. Basic spezifiziert die Signatur in PDF auf generellem Level. LTV ist spezifiziert die Long Time Validation.

2.2 Signatur von Dokumenten

Eine Signatur ermöglicht es den Urheber und die Integrität (Unmodifizierbarkeit) eines Dokumentes festzulegen. Die Signatur wird sozusagen über das Dokument gelegt. Die Signatur selber wird mit dem privaten Schlüssel des Signierenden und einem Hash (eine Art einzigartige Quersumme) des Inhalts erstellt. Mit dem öffentlichen Schlüssel des Signierenden kann der Hash wieder freigelegt werden und mit einem selbst erstellten Hash des erhaltenen Dokuments verglichen werden. Dabei wird die Identität des Urhebers und auch gleich die Integrität des Dokumentes bewiesen.
Die Signatur dient aber nicht zum Schutz des Inhalts (Verschlüsselung), sondern mehr als digitale Unterschrift.

Die Signatur von Dokumenten erfolgt in dieser Reihenfolge:

1. Dokument vorbereiten
2. Hash des zu signierenden Dokumentteils erstellen (also ohne den Signatur-Bereich)
3. Hash mit privatem Key signieren und Signatur daraus erstellen
4. Signatur in Dokument einbinden

[1] Weitere Informationen zu Lizenztypen unter http://www.gnu.org/licenses/license-list.html#apache2
[2] https://www.admin.ch/opc/de/classified-compilation/20131913/index.html
[3] https://www.bakom.admin.ch/bakom/de/home/digital-und-internet/digitale-kommunikation/elektronische-signatur.html
[4] https://www.bakom.admin.ch/bakom/de/home/das-bakom/organisation/rechtliche-grundlagen/vollzugspraxis/geraete-und-anlagen/normen.html

2.2.1 CMS

Die in PDF verwendete Syntax für die Signatur ist die 'Cryptographic Message Syntax'[5]. Im Profil PAdES Basic wird definiert wie die Signatur in einem PDF aufgebaut bzw. eingebunden wird.
CMS erstellt eine Struktur basierend auf ASN.1, was eine Art Baum mit diversen Datentypen ermöglicht. Eine detaillierte Beschreibung von CMS befindet sich in RFC 5652[6].

Nach PAdES Basic gibt CMS die optionale Möglichkeit zur Einbindung eines signierten Zeitstempels, dem eingebetteten Zeitstempel. Die Einbindung der ebenfalls optionalen 'revocation information' ist ebenfalls möglich, und ist vor allem für CAdES interessant. Da diese Bestandteil der signed Attributes sind, müssen diese vor dem Signieren eingebunden werden. Hierfür wurde in LTV eine andere Methode gestellt, die nachträgliche Einbindung der Validierung ermöglicht.

```
SEQUENCE  (2 elem)
   OBJECT IDENTIFIER  1.2.840.113549.1.7.2
   [0]  (1)
      SEQUENCE  (5 elem)
         INTEGER  1
         SET  (1 elem)
            SEQUENCE  (2 elem)
               OBJECT IDENTIFIER  2.16.840.1.101.3.4.2.1
               NULL
         SEQUENCE  (1 elem)
            OBJECT IDENTIFIER  1.2.840.113549.1.7.1
         [0]  (3)
            SEQUENCE  (3 elem)
               SEQUENCE  (8 elem)
                  [0]  (1)
                     INTEGER  2
                  INTEGER  (158 bit)
                  SEQUENCE  (2 elem)
                     OBJECT IDENTIFIER  1.2.840.113549.1.1.11
                     NULL
                  SEQUENCE  (3 elem)
                     SET  (1 elem)
                        SEQUENCE  (2 elem)
                           OBJECT IDENTIFIER  2.5.4.6
                           PrintableString  CH
```

Abbildung 1 Aufbau der ASN.1 Syntax einer CMS-Signatur.

Es ist auch möglich, dass mehrere Signierende in eine Signatur gepackt werden, was aber ein Spezialfall ist und nicht Bestandteil dieser Arbeit ist.

Anzumerken ist, dass einer Signatur die ganze Zertifikatskette des Signierenden angefügt werden sollte[7]. Dies ermöglicht eine schnellere Überprüfung der Signatur und zeigt die Identität des Signierenden.

2.3 Long Term Validation

Long Term Validation bedeutet eine Signatur so zu validieren, dass sie für lange Zeit valide bleibt. Das heisst, dass die Signatur gültig bleibt nachdem das Zertifikat abgelaufen oder sogar die CA ungültig ist. Hierfür wird eine Kombination aus signierten Zeitstempel und Validierung von Zertifikaten angewendet. Es müssen alle verwendeten Zertifikate validiert werden, damit keine Lücke entstehen könnte. Dazu gehören auch die Zertifikate des (nachträglichen) Zeitstempels.

Die Validierungsdaten sollen dem Dokument angefügt werden, damit alles an einem Ort bleibt und auch weit in der Zukunft überprüft werden kann. Es ist auch gut möglich, dass die Validierungsdienste nach der Signatur nicht mehr abrufbar sind.

[5] Beschrieben in Part 2: PAdES Basic, ETSI TS 102 778-2
[6] https://tools.ietf.org/html/rfc5652
[7] https://tools.ietf.org/html/rfc5652#section-10.2.3

Signierte Zeitstempel werden dazu verwendet, um den Zeitpunkt der Signatur bzw. deren Existenzzeitpunkt zu beweisen. Zur Sicherheit kann dieser auch den Zeitpunkt der Validierung sicherstellen.

2.3.1 LTV nach PAdES

Das *PAdES Long Term* Profil definiert eine Erweiterung des PDF-Standards für die Langzeitvalidierung einer PDF-Signatur.
Die Long-Term Validation für eine Signatur ist erst unter bestimmten Bedingungen gegeben.

- Über einen **signierten Zeitstempel** wird sichergestellt, dass die Signatur zu einem gewissen Zeitpunkt bereits existierte.
- Der DSS beinhaltet alle Validierungs-Informationen zu den in der Signatur vorhandenen Zertifikate.

Massgebend für die Beurteilung ob der LTV Zustand erreicht wurde, ist der Adobe Acrobat Reader. Dies aus zweierlei Gründen. Erstens wurde der Standard zu PDF und somit der Grundstein für PAdES-LTV von Adobe entwickelt und definiert (Und von ISO abgenommen). Zum anderen ist der Acrobat Reader praktisch die einzige Software, die die LTV-Fähigkeit einer Signatur richtig erkennt und anzeigt. Auch ist es möglich mit Acrobat die Validation und Zeitstempel nachträglich in das Dokument einzubinden. Unabhängig davon, woher das Dokument herkommt, solange dessen Zustand gültig ist.

Für die Entwicklung hat dies auch den Vorteil gehabt, dass mit Adobe eine Vorlage des Zielformats erstellt werden konnte, welche die Struktur für die einzelnen Bestandteile von LTV aufzeigte. Eine Erarbeitung des Verständnisses über die verschiedenen Standards war aber ebenso nötig um die einzelnen Probleme besser zu verstehen.

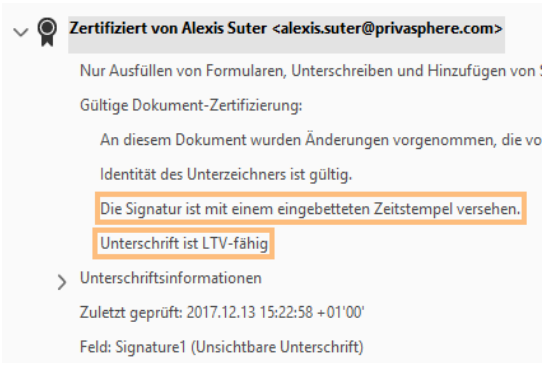


Abbildung 3 Adobe Acrobat zeigt auf ob eine Signatur LTV-fähig ist und wie der Zeitstempel eingebettet ist.

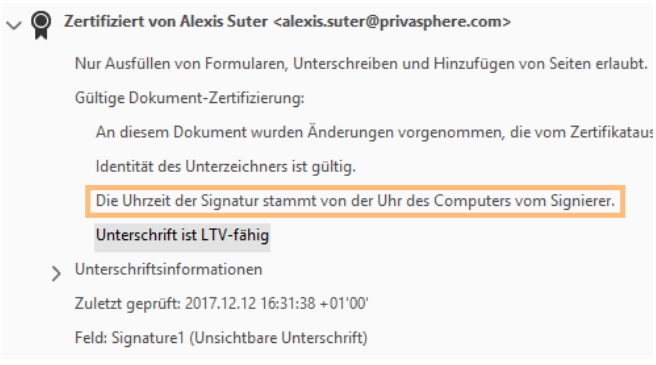


Abbildung 2 Adobe zeigt, dass die Uhrzeit vom Signierenden kommt.

Der Acrobat Reader zeigt aber nur ob die einzelnen Signaturen LTV-fähig sind. Die LTV-fähigkeit muss per dato manuell überprüft werden, was einiges am Verständnis der Standards verlangt, die im Folgenden genauer beschrieben werden.

2.3.1.1 LTV[8]

Für LTV wurde der DSS als eine Erweiterung von ISO 32000-1 (PDF) definiert. Der DSS lässt sich in den *Catalog* des Dokumentes einbinden. Zudem lässt sich dieser über Revisionen des Dokumentes erweitern. Dadurch entsteht die Möglichkeit zusätzliche Signaturen und Zeitstempel nachträglich in das Dokument einzubringen.
In Abbildung 4 wird gezeigt wie sich mehrere Signaturen und DSS verschachteln lassen. Diese Verschachtelung ermöglicht vorhergehende Inhalte durch Zeitstempel, der den bisherigen Inhalt umfasst, zu beschützen.

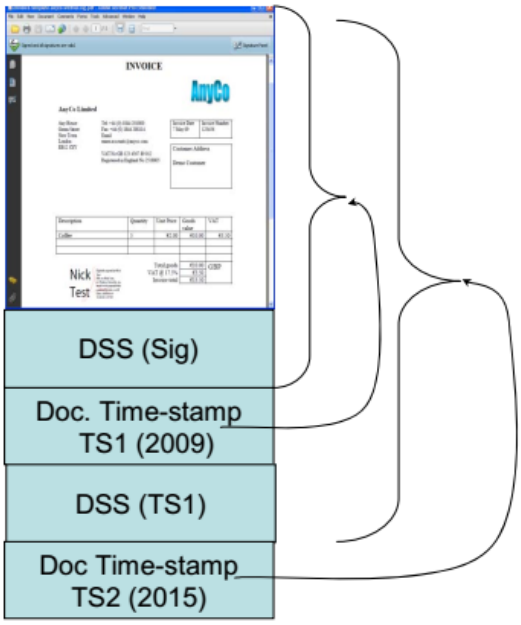
[8] Beschrieben in Part 4: PAdES Long Term, ETSI TS 102 778-4

Es ist allerdings nicht genau definiert, ob zuletzt ein DSS oder ein Zeitstempel stehen muss. Die Aussage hierzu aus Kapitel 4.1 aus PAdES LTV: *"The Document Time-stamp also protects the DSS by binding it to the document to which it applies."* Für maximale Sicherheit – was zwar etwas pedantisch ist – kann zuerst ein Dokument-Zeitstempel von TSA AlphaTime (fiktiv) eingebunden, dann der DSS zu diesem Zeitstempel eingebunden und schlussendlich nochmals ein Dokument-Zeitstempel der TSA AlphaTime eingebunden. Anderseits ist man dann auch nicht sicher ob die Validierung aus dem DSS zum Zeitpunkt des letzten Zeitstempels noch gültig ist.

> *"Furthermore, because the DSS is collected, and the signature first verified, at a time before the time indicated in the first Document Time-stamp, this indicated time can be used as the assumed signing time in a re-verification using the protected validation data from the DSS."*

Aus der Aussage vom selben Kapitel lässt sich ziehen, dass der erste Dokument-Zeitstempel da ist, um Zeitpunkt der gültigen – also Validen – Signatur zu bestimmen. Und der zweite Zeitstempel ist einfach da um den DSS zu beschützen.

Abbildung 4 aus PAdES Part 4, zeigt auf, wie ein Dokument mit mehreren Signaturen, Zeitstempeln und dazugehörigen DSS verschachtelt werden können.



2.3.1.2 Empfehlung zu LTV

Zusammengefasst benötigt LTV die Einbindung des DSS und des Dokumenten-Zeitstempel. In PAdES Part 4, Kapitel 4.2 wird empfohlen einen Zeitstempel über den DSS zu ziehen. Somit ist die Integrität des DSS sichergestellt.
Schlussendlich liegt es aber in den Händen der Anwender und Staaten, die genaue Umsetzung zu definieren.

2.4 PDF

Das Portable Document Format ist ein Dateiformat, das zum Ziel hat eine einheitliche Dartstellung von Dokumenten unabhängig von der Plattform zu ermöglichen. Der Standard für PDF 1.7 ist in ISO 32000-1 definiert. Für diese Arbeit ist nur diese Version von PDF im Einsatz.

Es wird nur grundlegend in die Details und Möglichkeiten von PDF eingegangen, da weiteres für diese Arbeit nicht nötig ist. Die Struktur – welche in einem Texteditor angesehen werden kann – von PDF wird ein wenig erläutert, um eine gewisse Vorstellung zu bekommen.

2.4.1 Revisionen / Inkremente

PDF Dokumente können erweitert werden, ohne das bisherige Dokument zu verändern. (ISO 32000-1

Kap. 7.5.6) Diese Revisionen können den Inhalt erweitern (auch überdecken) oder Metainformationen ergänzen.

2.4.2 Indirekte Objekte

Indirekte Objekte sind Objekte, die umfasst werden und können mit einer Referenznummer und weiteren Informationen wie Typ erweitert werden. (Für mehr: ISO 32000-1 Kap. 7.3.10)
Die Objekte, die mit einer ID versehen wurden, können dadurch immer wieder referenziert werden.

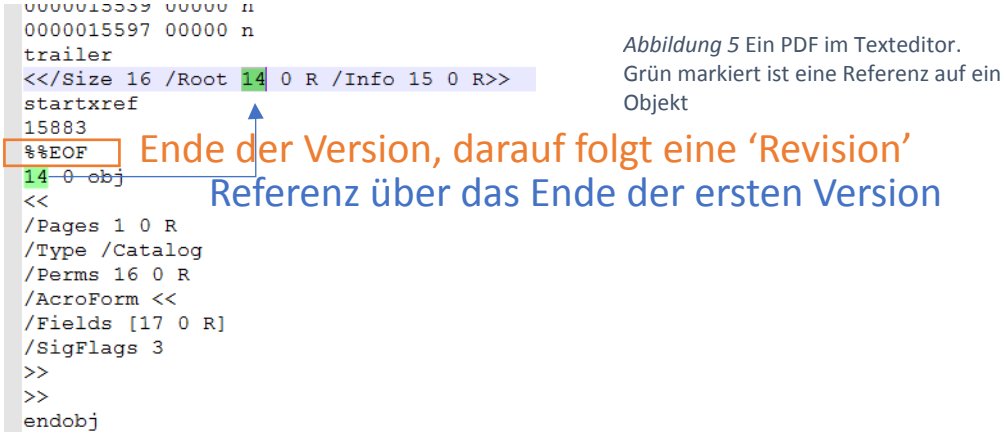


Abbildung 5 Ein PDF im Texteditor. Grün markiert ist eine Referenz auf ein Objekt

Es ist auch möglich deren Inhalt mit der gleichen ID in einer Revision zu überschreiben.

2.4.3 Objekttypen / Dictionary

Die definierten Objekttypen sind: Boolean values, Integer and Real numbers, Strings, Names, Arrays, Dictionaries, Streams, and the null object. (aus: ISO 32000-1 Kap. 7.3)

Ein sehr nützliches Objekt ist das Dictionary, welches Einträge mit Namen und von beliebigen Typ enthält. Damit lassen sich definierte Strukturen übersichtlich erstellen. Diese sind ähnlich einem OO-Objekt, nur lassen sie sich über Revisionen anpassen. Die PDF-Signatur ist zum Beispiel eine Anwendung davon.

2.4.4 Signaturen[9]

Signaturen werden in ein Dictionary gepackt, um diverse Informationen mit anzugeben. In nachfolgender Abbildung wird das Dictionary einer Signatur gezeigt. Die Signatur selber befindet sich im 'Contents'-Objekt statt dem **content**.

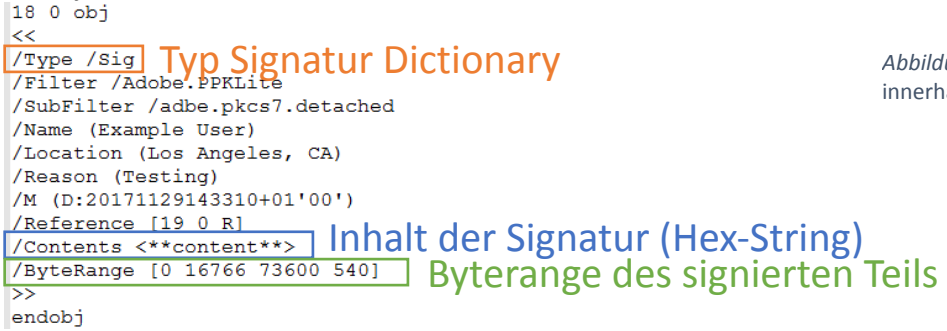


Abbildung 6 Signatur-Dictionary innerhalb eines PDFs

Die Byte Range ist nach dem Content das wichtigste Attribut. Diese besagt über welchen Teil des Dokumentes die Signatur gezogen wird. (1. Teil von bis, 2. Teil von bis) Die zwei mittleren Werte zeigen den Bereich des Contents der Signatur an (16766-73600).

Die Grösse des Contents muss im Vorherein festgelegt werden und kommt auf die Daten an, die in die Signatur eingebunden werden an.

[9] Definition unter: ISO 32000-1 (2008), Kapitel 12.8.1

2.4.5 PDFBox

Es ist Teil der Aufgabe, dass das Projekt auf Apache PDFBox aufbauen soll. PDFBox ist eine Java-Bibliothek zur Erstellung, Veränderung und Erweiterung von PDF Dokumenten. Für die Implementation der einzelnen Teilaufgaben werden neue Beispiele erstellt. Der Grund, dass man sich auf Beispiele beschränkt, ist dass einige Beispiele zu den Signaturen bereits vorhanden sind. Es gibt zudem bereits eine Implementierung für signierte Zeitstempel.

PDFBox stellt selber genügend Methoden für die Bearbeitung und Erweiterung der Dokumente. PDFBox übernimmt auch die Verwaltung der Objekte mit den Referenzen und Erweiterungen. Einzig die PDF-Signatur als Element ist fester Bestandteil der Library. Vieles rundum, wie das Signieren selber, das Timestamping usw. wird in den Beispielen entwickelt.

Zudem ist die Sicherheitsbibliothek BouncyCastle eingebunden, welche für die meisten sicherheitstechnischen Aufgaben nützlich ist. Von BouncyCastle können sehr viele Low-Level Aufgaben übernommen werden. Beispielsweise das Parsen von CMS / ASN.1, das Parsen von Zertifikaten etc. BouncyCastle wird vor allem beim Abruf der Zertifikats-Informationen und Validierungs-Daten behilflich sein.

Arbeiten, die Apache-Projekten beigesteuert werden, werden von erfahrenen Mitarbeitern überprüft und entweder zur Korrektur zurückgewiesen oder in den Code akzeptiert. Dies ermöglicht und erzwingt eine hohe Qualität des ausgelieferten Codes.

2.5 Timestamping

2.5.1 Signierter Zeitstempel (trusted timestamp)

Es gibt neben normalen Signaturen auch die Möglichkeit von extern signierten Zeitstempeln. Diese ermöglichen einen externen Beweis, dass ein Dokument zu einem gewissen Zeitpunkt genau den einen Zustand hatte. Im Unterschied zur herkömmlichen Signatur wird hier neben dem Hash noch ein Zeitstempel signiert und in der Signatur mitgegeben. Die nachfolgende Grafik zeigt den Ablauf zur Erstellung dieser Signatur vereinfacht dar.

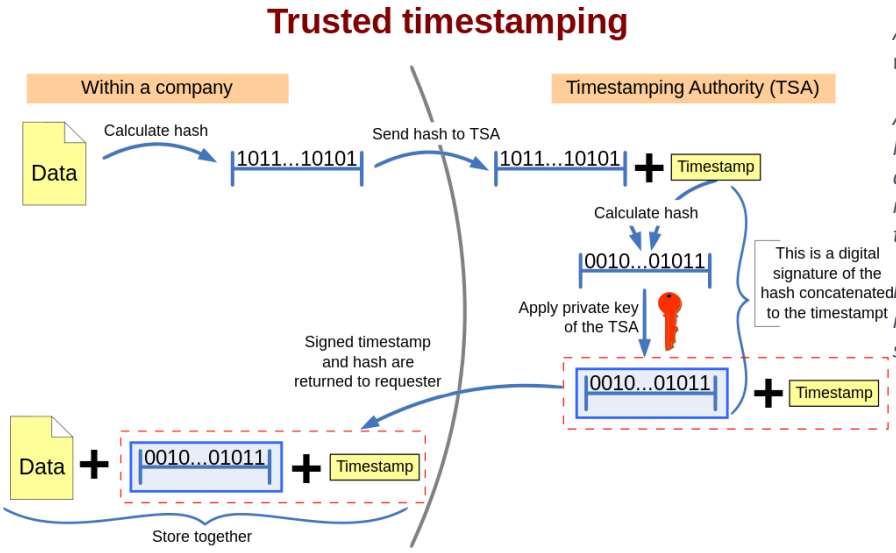


Abbildung 7 Das Estellen eines signierten Zeitstempels

Aus Wikipedia: https://commons.wikimedia.org/wiki/File:Trusted_timestamping.svg#/media/File:Trusted_timestamping.svg

Lizenz: https://creativecommons.org/licenses/by-sa/2.0/be/deed.en

2.5.1.1 Qualifizierter Zeitstempel / TSA

Wichtig ist, dass ein solcher Zeitstempel durch eine Timestamp Authority erstellt wird. Eine solche TSA muss einige Bedingungen erfüllen, die vom Bund bzw. in ETSI EN 319 421 geregelt werden. In (Schweizer) Gesetzen werden solche Zeitstempel als qualifiziert bezeichnet.

2.5.1.2 *Überprüfung des signierten Zeitstempels*

Die Überprüfung wird ähnlich der herkömmlichen Singnatur gemacht.

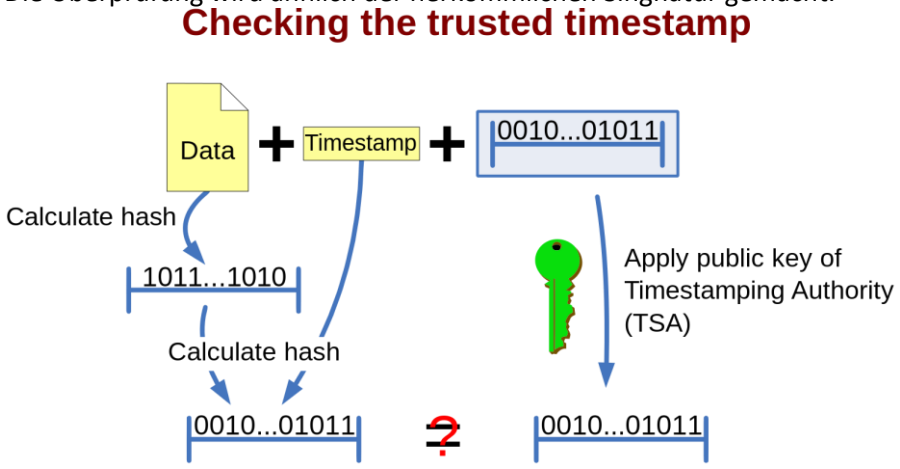


Abbildung 8 Das Überprüfen eines signierten Zeitstempels

aus Wikipedia, Lizenz: https://creativecommons.org/licenses/by-sa/2.0/be/deed.en

Quelle: https://commons.wikimedia.org/wiki/File:Checking_timestamp.svg#/media/File:Checking_timestamp.svg

2.5.2 Nachträglicher Zeitstempel

Um das Vorhandensein einer Signatur zu einem gewissen Zeitpunkt zu beweisen, wird ein signierter Zeitstempel eingesetzt. Nach dem Signieren wird ein signierter Zeitstempel erstellt und in das Dokument eingebunden. Hierfür gibt es zwei Möglichkeiten: dem umfassenden Zeitstempel und dem eingebetteten Zeitstempel.

Am relevantesten für die LTV ist der eingebettete Zeitstempel. Doch beide Vorgehen können benutzt werden und die erste Umsetzung belief sich auf den umfassenden.

Nachträglich ist der Zeitstempel, da dieser erst nach der Signatur erstellt wird. Vorher würde er nur den Inhalt zeitlich beschützen.

2.5.3 Umfassender Zeitstempel (Document Timestamp)[10]

Der umfassende Zeitstempel ist im PDF-Dokument der normalen Signatur sehr ähnlich. Er wird in "PAdES Part 4: PAdES LTV, Annex A.2" genauer beschrieben.

Der Hash wird dabei über den gesamten Inhalt des bisherigen Dokumentes – inklusive der vorbereiteten, angefügten Felder – gezogen. Daraus wird ein Antrag mit dem Hash an die TSA gesendet, welche den signierten Zeitstempel erstellt. Die Abbildung rechts zeigt auf, dass sich damit eine

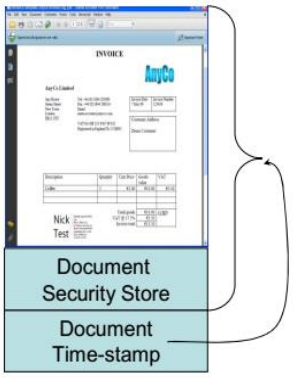


Abbildung 9 Dokument-Zeitstempel über bisherigen Dokumentteil – Aus PAdES Part 4

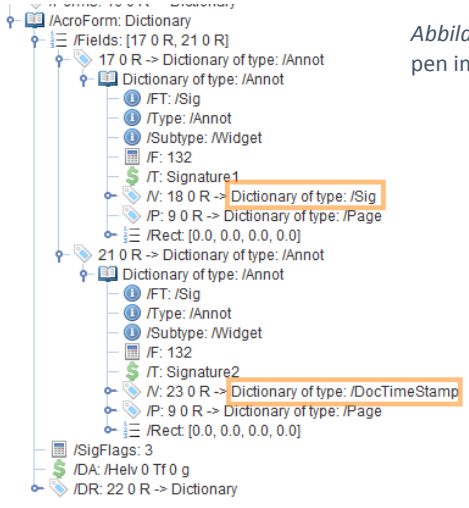


Abbildung 10 Die zwei Signaturtypen im Vergleich.

[10] Definition unter: ISO 32000-1 (2008), Kapitel 12.8.1

Verschachtelung von Zeitstempeln und weiteren (Validierungs-)Informationen erstellen lässt.

Nach dem diese Art von Einbindung entwickelt wurde, wurde allerdings klar, dass dies nicht wichtigster Bestandteil der LTV ist. Für weite Schritte wird diese Signatur aber durchaus brauchbar sein. Insbesondere wenn mehrere Stufen der Verschachtelung mit Validierung und Zeitstempeln stattfinden.

Dieser Teil war ein sehr guter Einstieg in das PDFBox Projekt und half einige Standards besser zu verstehen, insbesondere PDF und CMS.

2.5.4 Eingebetteter Zeitstempel[11]

Beim eingebetteten Zeitstempel ist das Vorgehen sehr unterschiedlich. Es geht darum, das Signaturfeld zu verändern. Dies mag gegensätzlich tönen, aber es gibt sogenannte Unsigned Attributes in den SignerInfo der CMS-Signatur[12]. Darin kann der Token eingefügt werden.[13] Diese Attribute sind nicht von der Signatur betroffen und können auch nachträglich geändert werden. Dabei kann die ASN.1 Baumstruktur zunutze gemacht werden um Daten einzufügen.

Voraussetzung für die eingebettete Signatur ist, dass genug Platz in der Signatur und dem Token gelassen wird. Bereits vor dem Signieren muss die Grösse des Signaturfeldes angegeben werden.[14] Das Feld wird dabei nach der Signatur mit ‘0’en auf die richtige Grösse aufgefüllt. Und auch deshalb wird im Vornherein eine genügende Grösse für den Signaturinhalt gewählt.

Für dieses Vorgehen wird ein Hash über die Signatur selbst aus den SignaturDaten gezogen. Dies bildet eine Art Gegen-Signatur, mit der der Zeitpunkt der Existenz sichergestellt wird. Wichtig bei diesem Vorgehen ist, dass keine weiteren Änderungen am vorhandenen Dokument entstehen und nur der Inhalt der Signatur erweitert wird.

2.6 Validierung von Signaturen

Um eine Signatur zu validieren, wird zum einen überprüft ob der Hash aus der Signatur mit dem Hash des Inhalts übereinstimmt. Dieses Vorgehen ist durch das Prinzip der Signatur immer gegeben. Zum anderen muss das Zertifikat des Signierenden auf dessen Gültigkeit überprüft werden.

Da Private Schlüssel gestohlen werden können, gibt es die Möglichkeit, beim Aussteller zu überprüfen, ob das Zertifikat noch gültig ist. Früher war vor allem CRL verbreitet, heute ist OCSP auf dem aktuellen Stand der Technik. Da bei OCSP eine aktuelle Antwort des Ausstellers erzwungen und auf ein Zertifikat hin spezifisch abgefragt werden kann, ist OCSP klar der CRL vorzuziehen.
CRL hat zudem den Nachteil sehr gross werden zu können, da sämtliche revozierte Zertifikate vom Aussteller auf einer Liste stehen.

OCSP wird in RFC6960 genau definiert.[15]

2.6.1 Validierung in LTV

Die Validierung der Signatur ist ein essenzieller Schritt für LTV. Das Vorgehen ist grob beschrieben:

1. Einsammlung der Zertifikat-Informationen aus der letzten Signatur
2. Abfrage der Validierungs-Informationen
3. Informationen in das Dokument einbinden

[11] Diese Methode wird in PAdES Part 2: Basic, Kapitel 4.3 beschrieben
[12] Welche in RFC 5652 beschrieben sind: https://tools.ietf.org/html/rfc5652#section-5.3
[13] Siehe Appendix A von RFC 3161: https://tools.ietf.org/html/rfc3161
[14] Definition unter: ISO 32000-1 (2008), Kapitel 12.8.1
[15] https://tools.ietf.org/html/rfc6960

OCSP wird in erster Linie benutzt, in zweiter CRL. Ein weiterer Grund ist, dass CRL wesentlich grösser sein kann und etliche, nicht benötigte Informationen enthält.

Für die Einbindung in das Dokument gibt es zwei Möglichkeiten:
Die erste Methode besteht darin, die Information in die Signatur einzubetten.
Für die zweite Methode wird das Dokument mit einem 'DSS' (Document Security Store) erweitert. Dieser enthält sämtliche Zertifikate, Ocsp-Antworten und wo nötig CRL-Antworten betreffend der letzten Signatur. Dieser Weg ist für die nachträgliche Validierung da.

2.6.2 Zertifikat-Findung und Zertifikat-Kette

Für die Validierung müssen zuerst sämtliche Zertifikat-Informationen der letzten Signatur gefunden werden. Dabei ist das Abarbeiten der Zertifikatskette massgebend. Normalerweise beinhaltet eine Signatur die gesamte nötige Zertifikatskette. Es gibt aber Fälle, in denen ein Issuer-(Herausgeber) Zertifikat aus dem Internet geladen werden muss. Vorsichtshalber wird dafür gesorgt, dass beide geladen und abgearbeitet werden.
Es werden diese Informationen aus den Zertifikaten gezogen:

- Aus der Erweiterung ' Authority Information Access'[16] werden diese Felder genommen:
  - OCSP-URL
  - Issuer-URL
- Aus der Erweiterung: 'CRL Distribution Points'[17]
  - Eine funktionierende CRL-URL
- Ob es Selbst-Signiert ist (Root-Zertifikat)
- Andernfalls das Issuer-Zertifikat.

2.6.3 Validierungs-Abfragen

2.6.3.1 OCSP

Wichtiges Feld für den OCSP-Request ist die CertID[18]. Dieses besteht aus Hashs des IssuerNamen und IssuerKeys, und der Seriennummer des Zertifikats. Dabei kann eine Nonce (Zufallszahl) mitgegeben werden, um die Einmaligkeit der Antwort zu erzwingen, da diese vom Server signiert wird.

Danach wird anhand der Antwort des Servers überprüft, ob das Zertifikat zum Abfragezeitpunkt noch gültig ist, also nicht wiederrufen wurde. Die Antwort wird dann ohne Bearbeitung in das Dokument eingebunden.

2.6.3.2 CRL

Steht kein OCSP-Server zur Verfügung oder gab es bei der Abfrage ein Problem (das Zertifikat ist aber nicht wiederrufen), kann auf CRL ausgewichen werden. Der Aufruf ist simpel, da einfach die URL abgefragt wird.
Die Antwort wird dann nach dem Zertifikat abgesucht. Ist dieses nicht in der Liste, so ist das Zertifikat noch gültig. Andernfalls wurde es wiederrufen.

2.6.4 Eingebettete Validierung[19]

Es gibt die Möglichkeit die Validierung, ähnlich wie beim Zeitstempel, in die Signatur einzubinden. Doch die Informationen werden in die Signed Attribute eingebunden. Was bedeutet, dass diese noch vor dem Signieren in die Signatur eingebunden werden müssen.

---

[16] Siehe RFC 5280 https://tools.ietf.org/html/rfc5280#section-4.2.2.1
[17] Siehe RFC 5280 https://tools.ietf.org/html/rfc5280#section-4.2.1.13
[18] Definiert in https://tools.ietf.org/html/rfc6960#section-4.1.1
[19] Beschrieben in Kapitel 12.8.3.3.2 von ISO 32000-1 (2008)

Dieser Ansatz wurde zwar begonnen, da aber die Aufgabe ist, nachträglich zu validieren, wurde er nicht mehr verfolgt. Jedoch ist der Grundstein für diese Implementierung bereits vorhanden, da die nötigen Informationen gesammelt wurden und die Lösung ähnlich dem eingebetteten Zeitstempel ist.

2.6.5 DSS[20]

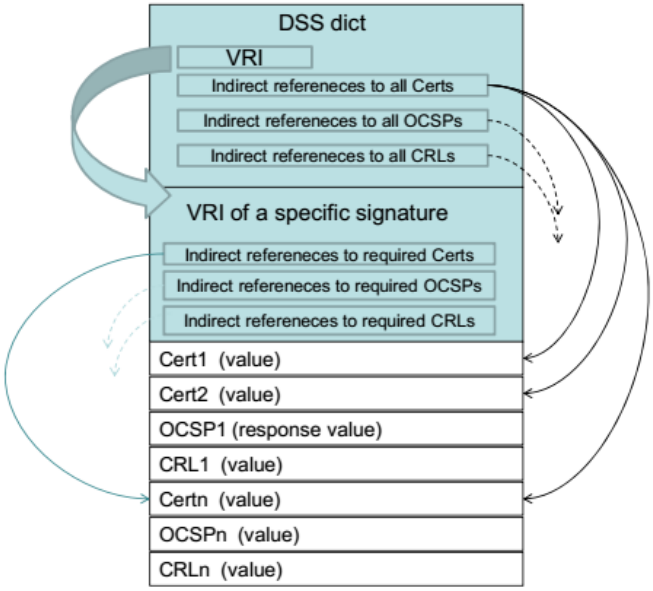


Abbildung 11 aus PAdES Part 4: LTV, Zeigt die Struktur des DSS und VRI Die Pfeile sind Referenzen auf die Inhalte (in Weiss).

Der Document Security Store, ist ein PDF-Dictionary, in dem diverse Informationen gespeichert werden können. Ein solches Dictionary wird in einer Art Baumstruktur aufgebaut. Die Daten werden in gleicher Form wie erhalten eingebunden, jedoch mit einem spezifischen Encoding. (Meist Flate-Decoding) Somit lassen sich die Daten wieder einfach einlesen und überprüfen.

Im DSS werden gespeichert:

- Certs: Beinhaltet alle Zertifikate der Signaturen – auch der signierte Zeitstempel und der OCSP-Antworten.
- OCSPs: Alle Antworten der OCSP-Server.
- CRLs: Alle CRL-Antworten, für die es keine Zertifikate gibt.
- VRI: Lexikon, welches die zugehörige Signatur zu dessen Validierungsinformation zuordnet.

Das VRI verknüpft die Validierungs-Daten mit der Signatur über einen Hash (Sha1[21]) der Signatur. Dabei wird ebenfalls auf die 3 Informationen verwiesen: Cert, OCSP und CRL. Da PDF über Referenzen arbeitet, ist es nicht nötig Daten doppelt aufzuführen.

3 Ergebnisse

Um die Ergebnisse der Arbeit aufzuzeigen wurden mehrere Beispiele erstellt, die die verschiedenen Stadien des Projektes aufzeigen.

Während der Arbeit gab es mehrere Tools zur Überprüfung der Resultate.

Zum einen zeigt der aktuellste **Adobe Acrobat Reader** sehr detaillierte Informationen zu den Signaturen. Sehr nützlich ist die Anzeige die zeigt, wie es um den Zeitstempel steht, und ob die Signatur LTV-fähig ist. Wenn die Signatur, oder anderes fehlerhaft ist, zeigt sich dies.

[20] Definiert in PAdES Part 4: LTV, Annex A.1
[21] Da dies nur ein Verweis ist, genügt Sha1 als Hash-Algorithmus

Ein weiteres Tool ist **iText RUPS**. Ein simples Tool, welches die innere Struktur des PDFs anzeigt. Mit einem Texteditor zusammen konnten einige Erkenntnisse über die Ziel-Struktur des DSS gewonnen werden.

Validator.ch ist eine Seite vom Bund, auf der man Dokumente bzw. deren Signatur überprüfen (validieren) kann. Es gibt diverse Signaturtypen, die überprüft werden können. Überprüft wird per dato aber nur die korrekte Signatur und den eingebetteten Zeitstempel, aber nicht die LTV-Fähigkeit. Es wird eine Art Checkliste abgegeben, die anzeigt, was mit der Signatur in Ordnung ist und was nicht.

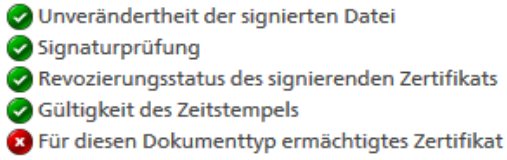


Abbildung 12 aus validator.ch Das Erreichen der einzelnen Schritte wird angezeigt. Das benutzte Zertifikat ist nicht für Unterschriften geeignet, daher die entsprechende Markierung. Z.B. eine SwissID wäre dafür geeignet.

3.1 Document Zeitstempel

Der erste grössere Schritt, der erreicht wurde, ist eine zusätzliche Signatur, die nur einen signierten Zeitstempel enthält. Das Vorgehen ist wie folgt:

1. Das zu signierende Dokument wird via PDFBox in ein Dokumentobjekt eingelesen.
2. Ein neues Signaturobjekt vom Typ 'DocTimeStamp' wird erzeugt und in das Dokument-Objekt eingefügt.
3. Das Dokumentobjekt wird neu geschrieben. Dabei wird die Byte-Range für die Signatur erstellt, die verdeutlicht, von wo bis wo Signiert wird. (Es sind zwei Bereiche, die die Signatur umschliessen)
4. Sobald die Struktur mitsamt Inhalt aufgebaut ist, wird das eigentliche Signieren durchgeführt. Hierzu wird der zu signierende Inhalt – über die angegebene ByteRange – übergeben um daraus die Signatur zu kreieren.
5. Die Signatur wird zurückgegeben und in das Dokument eingefügt.

Zur Veranschaulichung wurde ein bereits signiertes Dokument genommen. Abbildung 13 zeigt wie der Zeitstempel als Revision erscheint.
Anzumerken ist, dass das Dokument nach der ersten Signatur geändert wurde. Dies weil das Dokument mit der zweiten Signatur erweitert wurde. Ist die vorhergehende Signatur ungültig, wird dies angezeigt wie in Abbildung 14. Der Zeitstempel bleibt aber gültig, da dieser das Dokument umfasst, und für diese Signatur unverändert bleibt.
Weitere Details findet man in der inneren PDF-Struktur. Die 'normale' Signatur belegt zwar mehr Felder, die aber meist optional sind. Einzig ist eine Referenz auf MDP um die nachträgliche Manipulation des Dokumentes zu beinträchtigen. Weitere Details befinden sich in Abbildung 15.

Abbildung 13 (in Acrobat) Signatur und Zeitstempel in einem Dokument. Die zweite Signatur erscheint in einer Revision

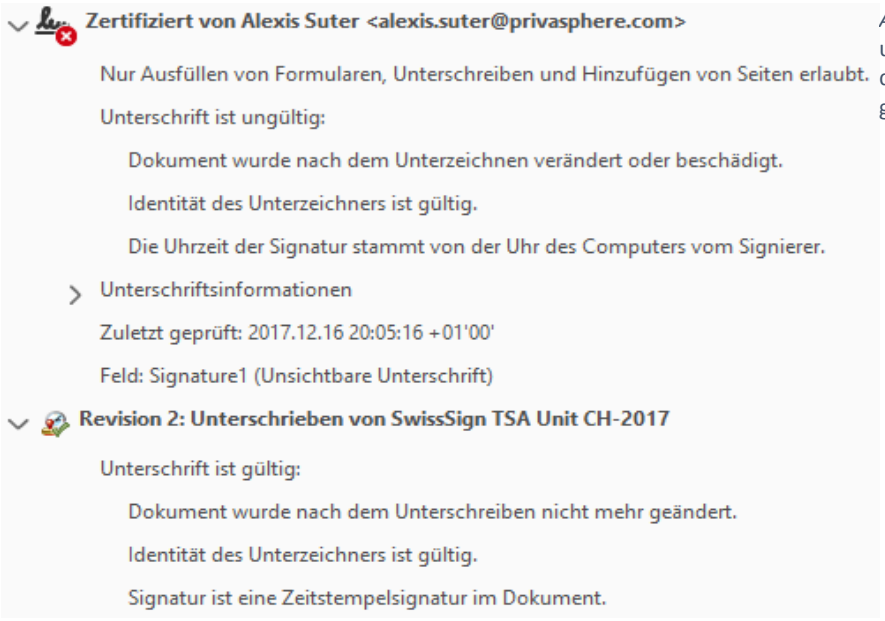


Abbildung 14 Die vorhergehende ungültige Signatur wird angezeigt, der Zeitstempel aber ist trotzdem gültig, was so stimmt.

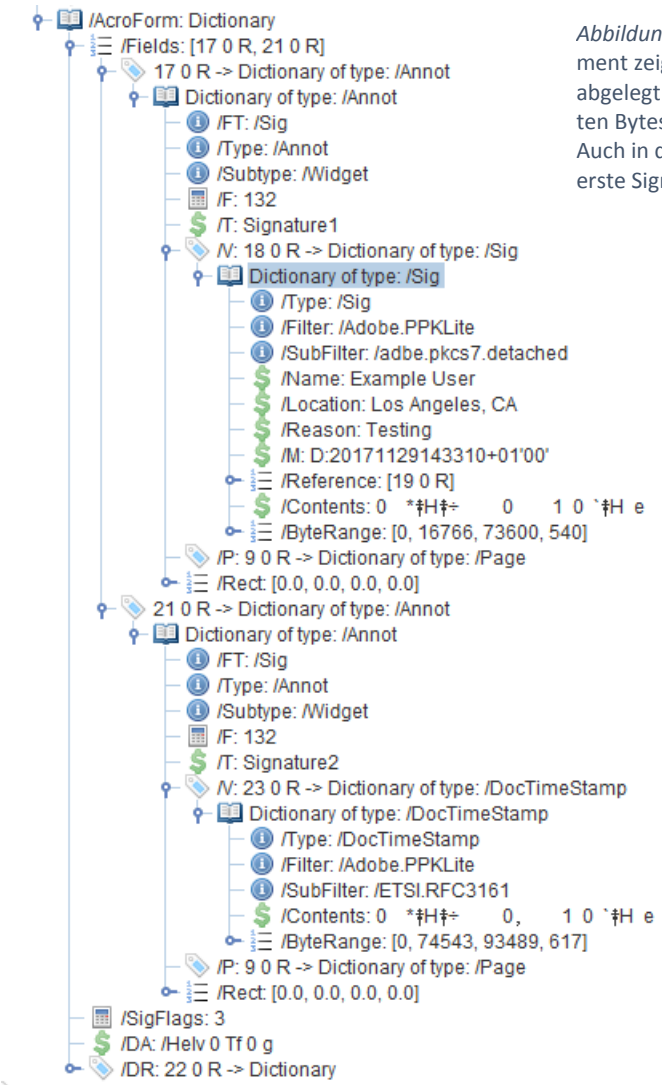


Abbildung 15 (aus iText) Die Struktur der Signaturen im Dokument zeigt wenige Unterschiede. Die Contents, wo die Signatur abgelegt werden, ist ebenfalls unterschiedlich, ausser in den ersten Bytes.
Auch in der ByteRange ist zu sehen, dass die zweite Signatur die erste Signatur und mehr einschliesst.

3.2 Eingebetteter Zeitstempel

Ein komplexerer Schritt ist die Einbindung eines Zeitstempels in die Signatur selbst. Es gibt zwei Möglichkeiten dies zu tun. Und zwar kann dies beim Signierens oder erst nach dem Signieren geschehen.
Für die erste Methode geht man folgendermassen vor: Zuerst wird die Signatur erstellt, dann der Zeitstempel über den Teil der Signatur erstellt, diesen in die Signatur eingebettet und dann in das Dokument eingebunden.
Bei der zweiten Methode ist das Vorgehen sehr ähnlich, es wird aber die existierende Signatur ausgelesen statt eine neue zu erstellen. Der Zeitstempel wird mit gleichem Vorgehen eingebettet.
Die erste Methode war bereits durch PDFBox gegeben. Für den nachträglichen Zeitstempel gab es aber noch keine Lösung. Hierfür wurde die bestehende Lösung angepasst, erweitert und optimiert; vor allem damit gewisse Codeteile nur einmal existieren.

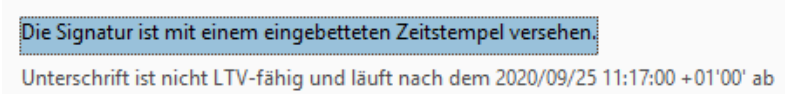


Abbildung 16 Der eingebettete Zeitstempel ist signiert.

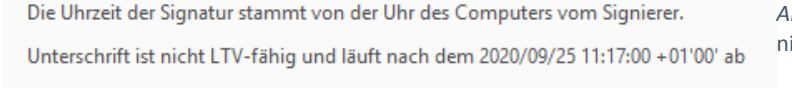


Abbildung 17 Die Urzeit in der Signatur ist nicht signiert.

Bei dieser Aufgabe wurde die ASN.1 Struktur intensiv analysiert, um besser zu verstehen wie CMS-Signaturen aufgebaut sind. Mit dabei war die Absicht auch die Validierungs-Daten in die Signatur einzubinden, was dann aber mangels Notwendigkeit für LTV übersprungen wurde.

3.3 Validierungsdaten

Um an die Validierungsdaten zu kommen werden diverse Schritte abgearbeitet:

- Extrahieren der letzten Signatur
- Extrahieren der Zertifikate des Signierenden

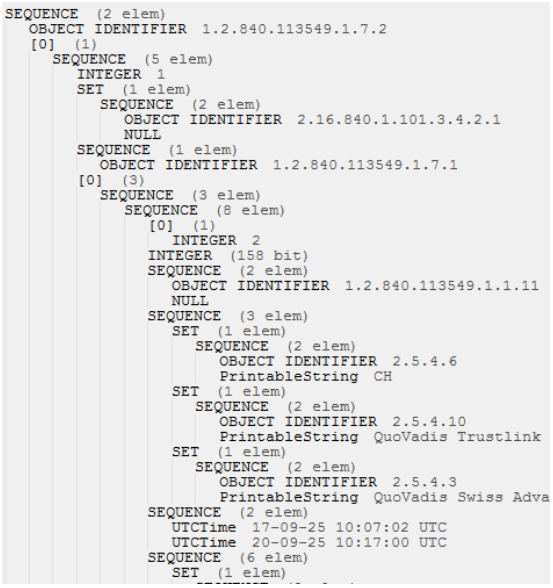


Abbildung 18 Die ASN.1 Struktur ist relativ komplex aufgebaut und mit Codes (OID's) verse-en um eine klare Struktur vorzugeben.

- Extrahieren der Zertifikate aus dem signierten Zeitstempel

3.3.1 Abfrage von OCSP

Um die OCSP Daten zu erhalten, muss eine korrekte Abfrage erstellt werden. Wichtiges Attribut ist die CertificateID, welche aus dem Issuer Zertifikat, und der Zertifikat-Seriennummer erstellt wird. Weiter wird eine Nonce eingebaut, um die Einmaligkeit der Antwort sicherzustellen.
Die OCSP Anfrage (die auf http basiert) wird dann durchgeführt und die Antwort weiter überprüft. Wichtig sind der Antwortstatus, der Status des Zertifikats und Überprüfung der Nonce.
Wenn das Zertifikat revoziert wurde, wird die Validierung abgebrochen. Ist die Antwort ungültig oder

Tritt ein Fehler auf, wird auf CRL ausgewichen. Falls CRL ebenfalls nicht verfügbar ist, wird die Validierung abgebrochen, da die Gültigkeit des Zertifikats nicht festgestellt werden kann.

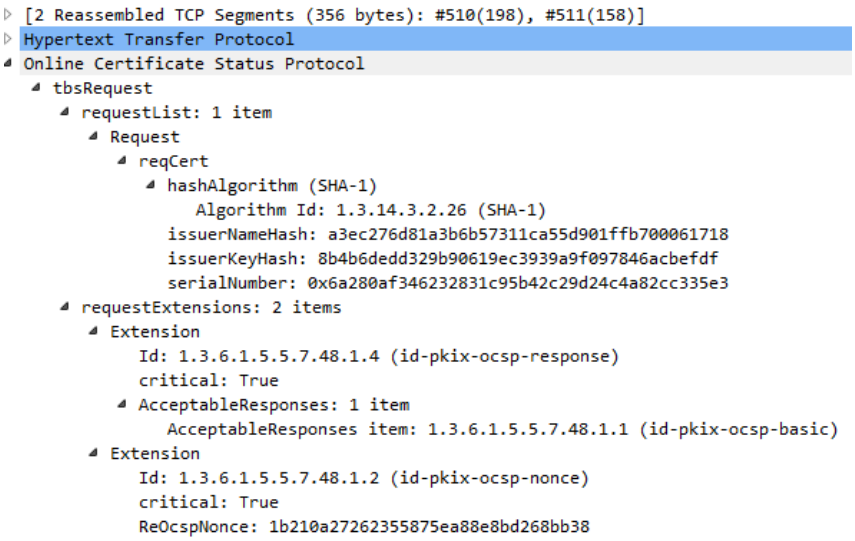


Abbildung 19 Die OCSP-Anfrage mit einer Nonce und der CertificateID, bestehend aus issuer-NameHash/-KeyHash und serialNumber.

Abbildung 20 Die OCSP-Antwort mit selber Nonce und CertificateID. Angehängt ist auch das Zertifikat mitsamt Signatur des Ausstellers.

3.3.2 Abfrage von CRL

Die Abfrage von CRL gestaltet sich wesentlich einfacher, da im Prinzip nur die gegebene URL abgefragt werden muss. Dann wird überprüft, ob das angegebene Zertifikat auf der Liste ist. Falls ja, ist das Zertifikat revoziert und die Validierung wird abgebrochen.

3.4 DSS

Die gesammelten Daten werden darauf in die Dokumentstruktur eingebunden. Da vieles auf den gleichen Standards basiert, können die Daten wie Zertifikate, OCSP-Antwort und CRL-Antworten relativ einfach kodiert eingebunden werden.
Die Daten werden in einer Struktur verlinkt, die hilft die Daten einfacher zu finden.
Es gibt auch die Möglichkeit den DSS über Revisionen zu erweitern. Dafür wird via PDFBox dasselbe Objekt genommen, wird zum Updaten markiert und mit entsprechenden Einträgen erweitert. Im neuen Dokument erscheint dann wider der gesamte DSS, die Daten bleiben aber weiter nur verlinkt und müssen nicht mitkopiert werden.

3.4.1 Benutztes Zertifikat

Um diverse Tests zu machen wurde meist das gleiche Zertifikat, signiert durch QuoVadis, benutzt.

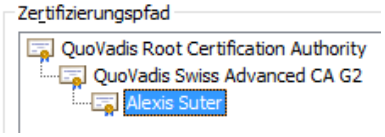
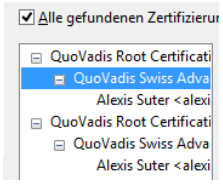


Abbildung 21 Das benutzte Zertifikat und seine Zertifikatkette

Abbildung 22 Alternative Zertifikatpfade in Adobe



Bei der Erarbeitung stellte sich heraus, dass ein Zertifikat mehrere Issuer-Pfade haben kann. Diese haben zwar den gleichen Public Key, aber einen anderen Fingerabdruck / Seriennummer diese wird durch

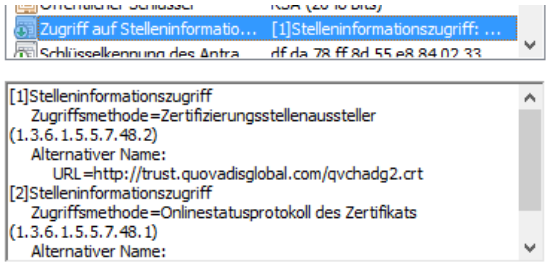


Abbildung 23 Eintrag zum alternativen Zertifikat neben der OCSP-URL

den Adobe Reader angezeigt, welcher beide Zertifikate erkennt. Der Eintrag zum alternativen Zertifikat befindet sich in den Attributen des Zertifikats.

3.4.2 DSS nur für eine Signatur

Um ein Beispiel aufzuzeigen, siehe Abbildung 24. Der VRI der Signatur zeigt dabei auf 3 OCSPs. Eines für den Signierenden und zwei für die Issuer-Zertifikate. Wobei der eine Issuer aus dem Zertifikat selbst kommt und der zweite über die alternative URL geladen wurde.

```
Got OCSP for EMAILADDRESS=alexis.suter@privasphere.com, CN=Alexis Suter,
O=PrivaSphere AG, L=Zürich, ST=ZH, C=CH
Got OCSP for CN=QuoVadis Swiss Advanced CA G2, O=QuoVadis Trustlink
Switzerland Ltd., C=CH
Got OCSP for CN=QuoVadis Swiss Advanced CA G2, O=QuoVadis Trustlink
Switzerland Ltd., C=CH
```

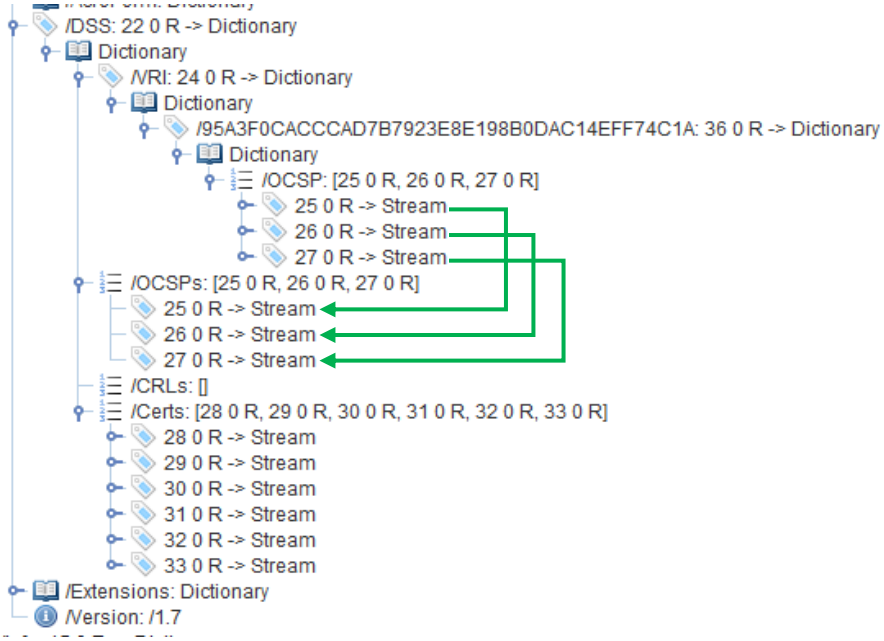


Abbildung 24 Die Visualisierung des PDFs zeigt die Referenzen auf die OCSP-Daten auf. Das VRI besitzt einen Eintrag für den Signierenden, welcher wiederum auf 3 OCSP-Einträge zeigt, die ebenfalls im DSS (OCSPs) direkt referenziert sind.

3.4.3 DSS für signierten Zeitstempel

Für die gewählte Zeitstempelautorität (TSA) ergibt sich ein Fehler bei der Abfrage des OCSPs. Das Ausweichen auf CRL funktioniert dabei und es folgt daraus ein anderes DSS.

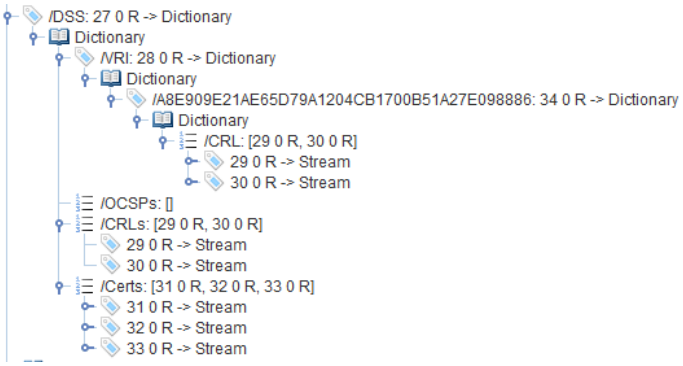


Abbildung 26 DSS nur vom Zeitstempel und nur mit CRL

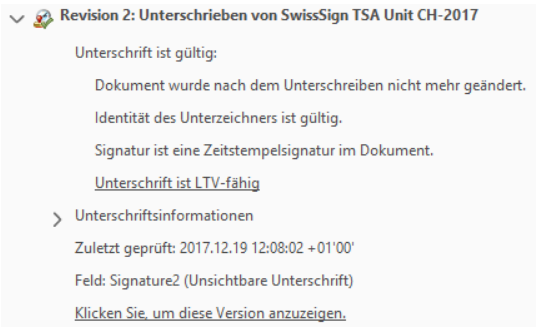


Abbildung 26 Der Zeitstempel ist wegen des DSS LTV-fähig

3.4.3.1 MALFORMED_REQUEST

Der Auftretende Fehler bei der OCSP-Anfrage an die TSA, wird von dieser als falsch gebildete Anfrage gewertet. Beim Weglassen der Attribute (also auch der Nonce) funktioniert die Anfrage. Ein Rückfall auf CRL ist aber richtig, da dies als Ausnahmefall gewertet wird.

3.4.4 DSS über zwei Signaturen

Das nächste Beispiel wurde mit einem signierten Dokument begonnen, über das der erste DSS erstellt wurde. Darauf mit einer Zeitstempelsignatur versehen und ebenfalls mit einem DSS versehen. Daraus entsteht dann einer Erweiterung des DSS, welcher auch durch die Sprünge in den Referenznummern sichtbar wird.

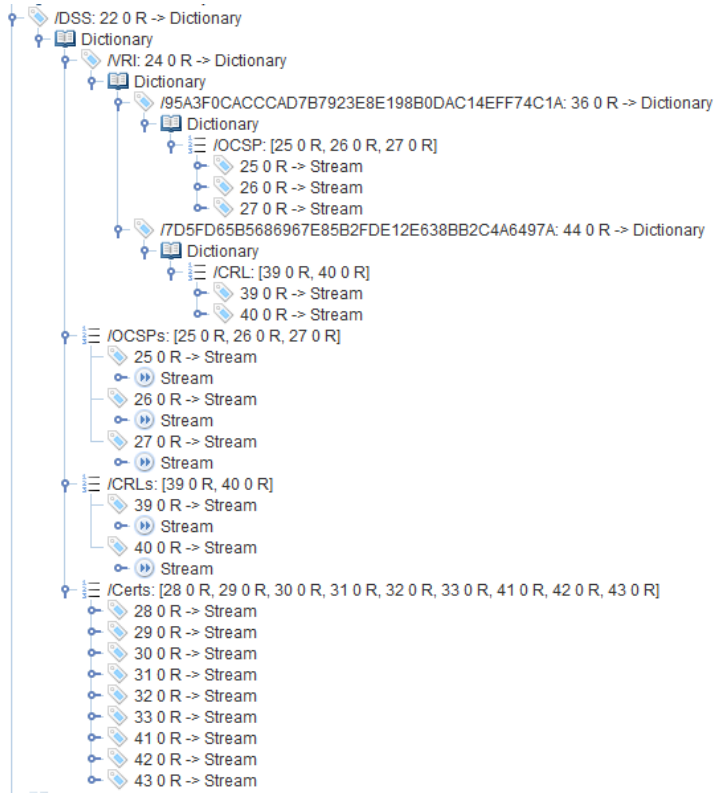


Abbildung 27 DSS über zwei Signaturen (eine Regulär und ein signierter Zeitstempel). Die Sprünge sind in den Zertifikaten ersichtlich (33 und 41) und zwischen den OCSP der regulären Signatur und den CRL des signierten Zeitstempels.

3.4.5 DSS über Signatur mit eingebettetem Zeitstempel

Wird der DSS über eine Signatur mit eingebettetem Zeitstempel gemacht, bleibt man auf einen VRI beschränkt. Referenziert wird nur die ganze Signatur (der Hash) und darin auf die Validierung des signierenden Zertifikats. Trotzdem wird zu allen Zertifikaten die Validierung geholt und im DSS mit den Zertifikaten abgelegt.

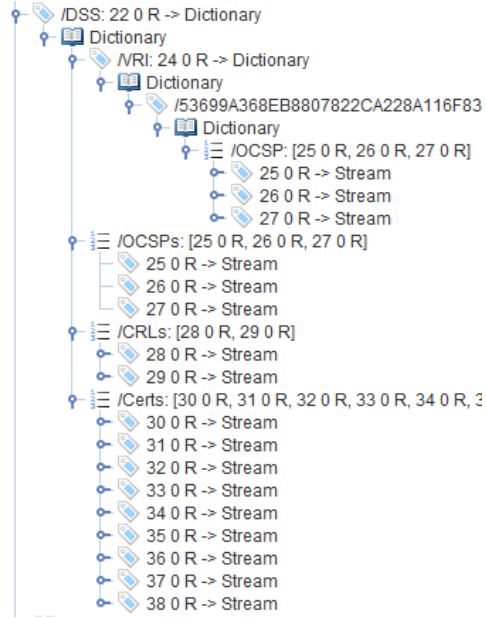
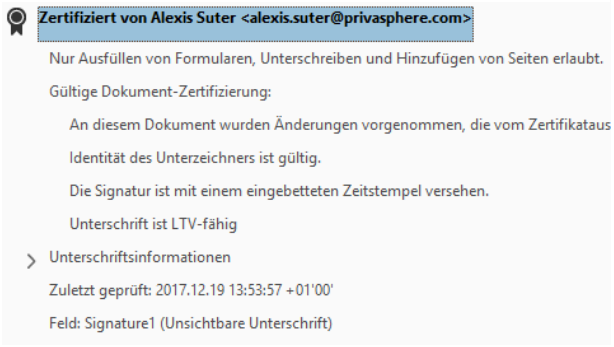


Abbildung 29 Der DSS über Signatur und eingebetteten Zeitstempel

Abbildung 29 Diese Signatur ist in Adobe bereits LTV-fähig



3.5 validator.ch

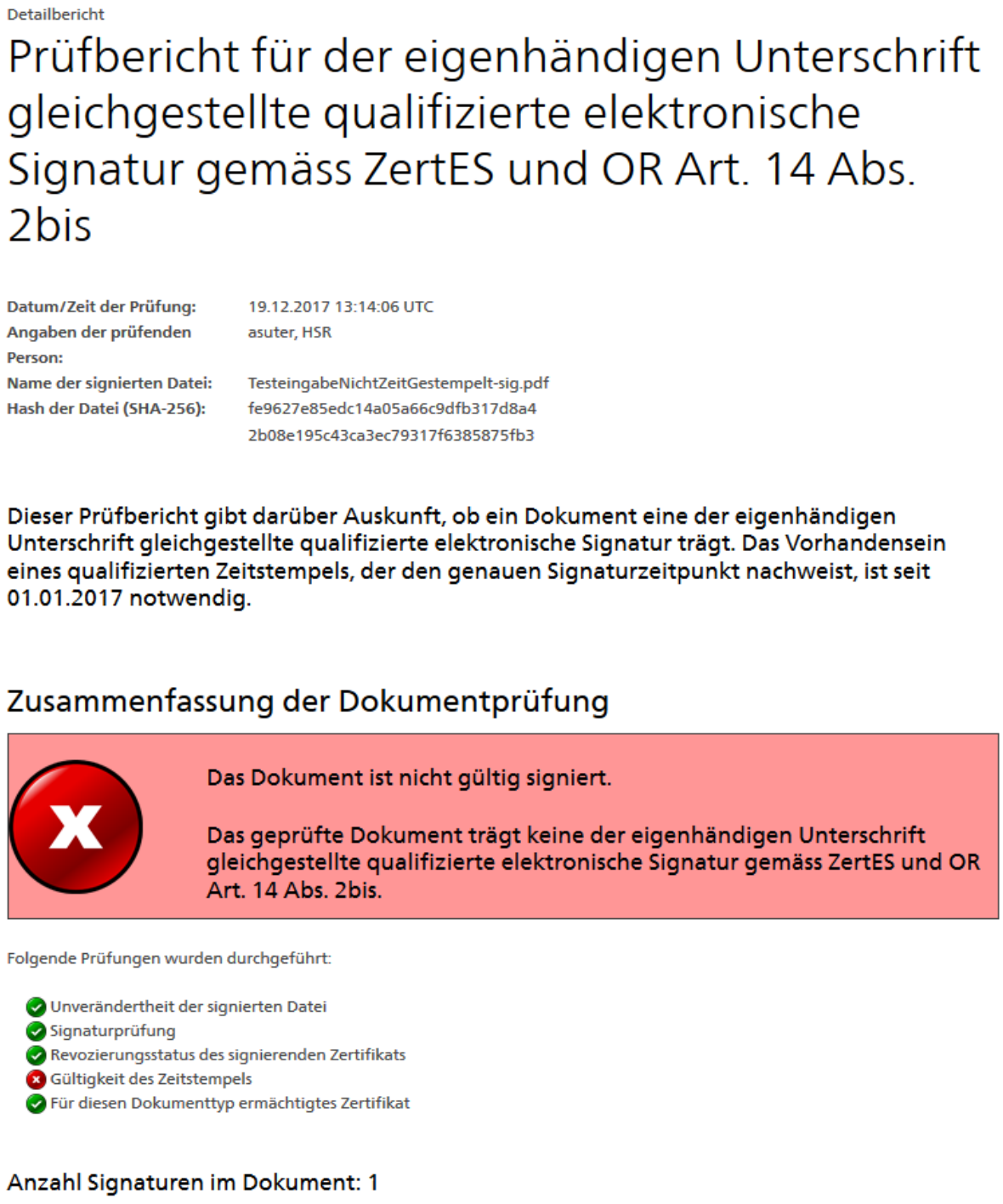
Dieses Kapitel soll mehrere Aspekte und Resultate von validator.ch aufzeigen. Validator.ch ist ein Dienst, der vom Bund zur Verfügung gestellt wird um signierte Dokumente zu validieren. Das Ziel ist, visuell klar zu verdeutlichen, ob ein Dokument gültig signiert ist. Da es sich um einen Online-Dienst

handelt, beschränkt sich der Bericht auf die Version, die am 19.12.2017 aufgeschaltet war.

Wichtig ist, dass validator.ch nicht auf LTV-fähigkeit überprüft. Es werden lediglich eine Signatur und dessen Zeitstempel überprüft.

3.5.1 Prüfbericht

Der Prüfbericht ist sehr umfangreich und zeigt einige Details zur Signatur auf. Das nachfolgende Beispiel eines Berichts wurde über ein signiertes Dokument gemacht. Das benutzte Zertifikat ist für die digitale Signatur ermächtig (wie z.B. SwissID). Doch für die Validität fehlt noch der signierte Zeitstempel.



Detailbericht

Prüfbericht für der eigenhändigen Unterschrift gleichgestellte qualifizierte elektronische Signatur gemäss ZertES und OR Art. 14 Abs. 2bis

Datum/Zeit der Prüfung:	19.12.2017 13:14:06 UTC
Angaben der prüfenden Person:	asuter, HSR
Name der signierten Datei:	TesteingabeNichtZeitGestempelt-sig.pdf
Hash der Datei (SHA-256):	fe9627e85edc14a05a66c9dfb317d8a4 2b08e195c43ca3ec79317f6385875fb3

Dieser Prüfbericht gibt darüber Auskunft, ob ein Dokument eine der eigenhändigen Unterschrift gleichgestellte qualifizierte elektronische Signatur trägt. Das Vorhandensein eines qualifizierten Zeitstempels, der den genauen Signaturzeitpunkt nachweist, ist seit 01.01.2017 notwendig.

Zusammenfassung der Dokumentprüfung

Das Dokument ist nicht gültig signiert.

Das geprüfte Dokument trägt keine der eigenhändigen Unterschrift gleichgestellte qualifizierte elektronische Signatur gemäss ZertES und OR Art. 14 Abs. 2bis.

Folgende Prüfungen wurden durchgeführt:

- Unveränderheit der signierten Datei
- Signaturprüfung
- Revozierungsstatus des signierenden Zertifikats
- Gültigkeit des Zeitstempels
- Für diesen Dokumenttyp ermächtigtes Zertifikat

Anzahl Signaturen im Dokument: 1

Prüfdetails Signatur 1

Zeitpunkt der Unterschrift: 25.10.2017 16:44:41 UTC
Signaturalgorithmus: SHA256
Die digitale Signatur ist gültig (Details siehe A)

Information über den Zeitstempel

Der Signaturzeitpunkt wurde nicht durch einen authorisierten TSA (Time Stamp Authority) bestätigt.

Information über das Unterzeichnerzertifikat

Zertifikat ausgestellt für:	Ralf Christian Hauser-Thoma (Qualified Signature) (hauser@privasphere.com)
Zertifikat ausgestellt von:	SwissSign Qualified Platinum CA 2010 - G2 (SwissSign AG)
Gültigkeit des Zertifikats:	11.12.2016 bis 11.12.2019
Revokationsstatus:	Zertifikat nicht revoziert
Zertifikatsträger:	Hardware Token
Zertifikatsklasse:	Zertifikat erlaubt die Erstellung qualifizierter elektronischer Signaturen

Prozessbezogene Prüfung

Validator:	Der eigenhändigen Unterschrift gleichgestellte qualifizierte elektronische Signatur
Prüfung:	Das Zertifikat ist ein qualifiziertes Zertifikat einer anerkannten Anbieterin gemäss ZertES.

Gültigkeit einer Signatur:

(A) Eine gültige Signatur besitzt folgende Eigenschaften:

- Alle Zertifikate in der Signatur wurden mathematisch geprüft.
- Es ist sichergestellt, dass der Unterzeichner den Schlüssel seines Zertifikats für die Signatur verwendete.
- Der Zertifikatspfad jedes Zertifikats wurde geprüft. Dadurch wird die Echtheit des Zertifikats des Unterzeichners durch unabhängige, vertrauenswürdige Zertifikate bestätigt.
- Das Zertifikat des Unterzeichners sowie alle übergeordneten Zertifikate des Ausstellers waren zum Zeitpunkt der Signatur gültig.

3.5.2 Bericht mit Dokument-Zeitstempel

Der externe Zeitstempel wird vom Dienst nicht erkannt. Ein wichtiger Punkt ist, dass das Dokument als verändert markiert wird. In der Zeitstempelsignatur wird zudem kein Zeitstempel erkannt.

Zusammenfassung der Dokumentprüfung

Das Dokument ist nicht gültig signiert.

Das geprüfte Dokument trägt keine der eigenhändigen Unterschrift gleichgestellte qualifizierte elektronische Signatur gemäss ZertES und OR Art. 14 Abs. 2bis.

Folgende Prüfungen wurden durchgeführt:

- ✗ Unverändertheit der signierten Datei
- ✗ Signaturprüfung
- ✓ Revozierungsstatus des signierenden Zertifikats
- ✗ Gültigkeit des Zeitstempels
- ✗ Für diesen Dokumenttyp ermächtigtes Zertifikat

Abbildung 30 Einzig das Zertifikat des Zeitstempels wird erkannt / akzeptiert.

Anzahl Signaturen im Dokument: 2

Information über den Zeitstempel

Der Signaturzeitpunkt wurde nicht durch einen authorisierten TSA (Time Stamp Authority) bestätigt.

Information über das Unterzeichnerzertifikat

Zertifikat ausgestellt für:	SwissSign TSA Unit CH-2017
Zertifikat ausgestellt von:	SwissSign TSA Platinum CA 2017 - G22 (SwissSign AG)
Gültigkeit des Zertifikats:	17.02.2017 bis 17.02.2028
Revokationsstatus:	Zertifikat nicht revoziert

Abbildung 31 Das akzeptierte Zertifikat für den Zeitstempel

3.5.3 Bericht mit eingebettetem Zeitstempel

Wird in dasselbe signierte Dokument der Zeitstempel aber eingebettet, zeigt sich validator.ch wesentlich positiver und markiert die Signatur als gültig.

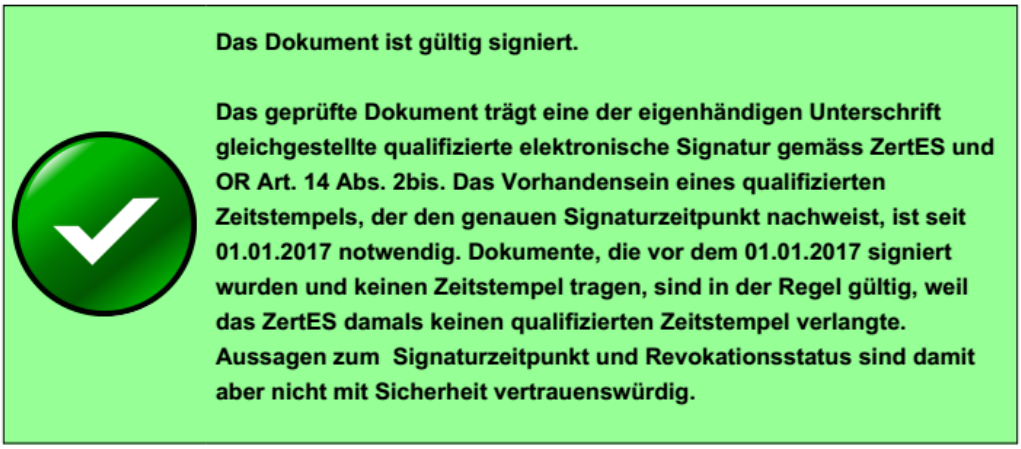


Abbildung 32 Die klare Anzeige, dass ein Dokument gültig signiert wurde.

- Unveränderheit der signierten Datei
- Signaturprüfung
- Revozierungsstatus des signierenden Zertifikats
- Gültigkeit des Zeitstempels
- Für diesen Dokumenttyp ermächtigtes Zertifikat

Information über den Zeitstempel

Zertifikat ausgestellt für:	SwissSign TSA Unit CH-2017
Zertifikat ausgestellt von:	SwissSign TSA Platinum CA 2017 - G22 (SwissSign AG)
Gültigkeit des Zertifikats:	17.02.2017 bis 17.02.2028
Der Zeitstempel ist gültig	

Abbildung 33 links: Alle Punkte von validator.ch sind erfüllt. *rechts:* Der Zeitstempel mit seiner Signatur ist gültig

3.5.4 Dokument mit DSS

Wird das gültig signierte Dokument mit einem DSS versehen sieht der Bericht etwas anders aus, bleibt aber gültig. Die Erweiterung durch den DSS verändert das Dokument selber nicht und trägt im Grunde nur Hintergrundinformationen. Somit wird hierzu ein Hinweis angezeigt und die Signatur bleibt gültig.

Folgende Prüfungen wurden durchgeführt:

- Unveränderheit der signierten Datei
- Signaturprüfung
- Revozierungsstatus des signierenden Zertifikats
- Gültigkeit des Zeitstempels
- Für diesen Dokumenttyp ermächtigtes Zertifikat
- Signatur deckt ganzes Dokument ab. Die Änderungen nach der letzten Signatur werden als erlaubt eingestuft.

Abbildung 34 Mit einem qualifizierten Zertifikat signiertes Dokument.

Anzahl Signaturen im Dokument: 1

3.5.5 Zusätzliche Signatur

Es ist auch möglich ein zweifach signiertes Dokument zu validieren. Aber beide Signaturen müssen einen signierten Zeitstempel beinhalten. Was aber heisst, dass die erste Signatur vor der zweiten mit einem solchen versehen werden muss.

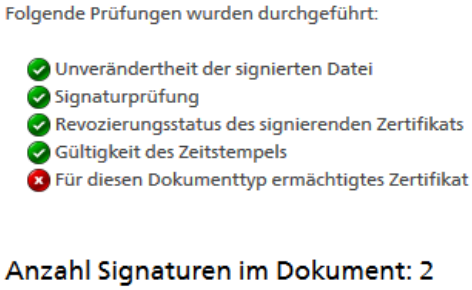


Abbildung 35 Zwei gültige Signaturen in einem Dokument

4 Implementation

Weiter soll in die Implementation der Beispiele eingegangen werden. Auch Anpassungen des Codes werden beschrieben.
Der Code ist ersichtlich unter:
https://svn.apache.org/repos/asf/pdfbox/trunk/examples/src/main/java/org/apache/pdfbox/examples/signature/ und im Unterverzeichnis für die Validierung:
https://svn.apache.org/repos/asf/pdfbox/trunk/examples/src/main/java/org/apache/pdfbox/examples/signature/validation/

4.1 Zeitstempel

Für die Zeitstempel musste wenig neuer Code erstellt werden, da bereits einiges in CreateSignature/-Base für das Timestamping umgesetzt war. Die drei □-markierten Klassen wurden neu erstellt. □ wurde erweitert.

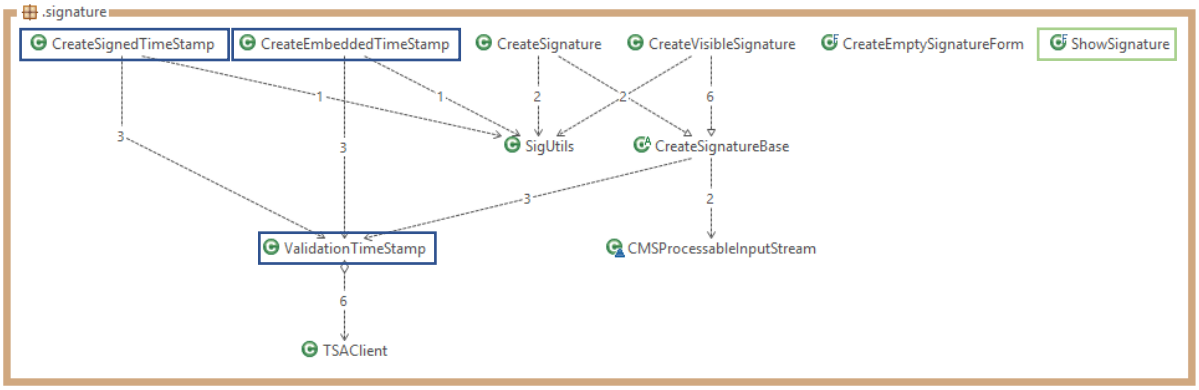


Abbildung 36 Die Klassen für die verschiedenen Signiermethoden (und ihre Abhängigkeiten).

ValidationTimeStamp sammelt Methoden um TSA-Anfragen zu erstellen und die Antworten in entsprechende Strukturen einzusetzen. Es gibt soweit zwei Anwendungen: Die eine ist das Einbetten des Zeitstempels in eine Signatur, was relativ komplex ist. Die andere gibt den signierten Zeitstempel unbearbeitet zurück, um diesen als Signatur einzubinden.
CreateSignedTimeStamp erstellt einen signierten Zeitstempel über das bisherige Dokument. Das Prinzip ist relativ simpel und geht Grundsätzlich gleich wie bei normalen Signaturen vor (z.B. CreateSignature).
CreateEmbeddedTimeStamp fügt einer vorhandenen Signatur einen signierten Zeitstempel an. Dabei wird die letzte Signatur im Dokument geholt, und der Klasse ValidationTimeStamp übergeben. Darin werden dann die 'unsigned attributes' mit einem Zeitstempel versehen. Die alte Signatur wird dann in einer Kopie des Dokuments mit der neuen überschrieben.
ShowSignature wurde erweitert um den DSS zu analysieren und dessen Inhalte anzuzeigen.

4.2 Validierung

Für die Validierung musste sämtlicher Code neu geschrieben werden. Die nötigen Informationen zu den Zertifikaten werden vom **CertInformationCollector** gesammelt. In **AddValidationInformation** werden diese Informationen Verarbeitet, die Revocation-Informationen abgefragt und in ein DSS eingebunden.

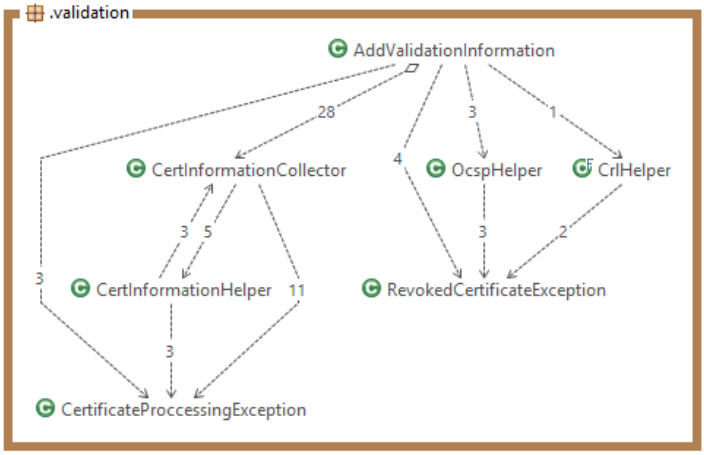


Abbildung 37 Die Klassen zur Validierung und Erstellung des DSS.

4.2.1 Revocation Helpers

Der **OcspHelper** übernimmt sämtliche Aufgaben zur Überprüfung des Zertifikat-Status via OCSP. Die meisten Arbeiten werden mit Hilfe von BouncyCastel durchgeführt. Unterteilt wird in die drei Teilaufgaben: Anfrage vorbereiten, Anfrage durchführen und Antwort überprüfen. Überprüft wird:

- der Antwortstatus (ob die Anfrage gültig ist)
- ob die Nonce in der Antwort stimmt
- ob das Zertifikat gültig ist
- ob das Zertifikat noch gültig ist

CrlHelper ist wesentlich simpler. CRL kann einfach über eine URL mittels http abgefragt werden. Die Antwort wird geparst und es wird überprüft ob das Zertifikat nicht auf der Liste steht.

4.2.2 Zertifikat Informationen einsammeln

4.2.2.1 CertInformationCollector

Um an sämtliche Informationen einer Zertifikatskette zu gelangen sind einige Schritte nötig.
Der grösste Teil der Arbeit wird in **CertInformationCollector** erledigt. Hierzu eine grobe Aufzählung der Schritte, die ein zwei Teile eingeteilt und bewusst redundant beschrieben ist. Zuerst über die gesamte Signatur:

1. Signatur aus Dokument holen, in CMS Objekt parsen.
2. Den Signierenden ausfindig machen und das Zertifikat *abarbeiten*
3. Dessen AuthorityInfoAccess überprüfen auf Issuer-Zertifikate aus dem Netz und dieses abarbeiten
4. Den Issuer, welcher im Zertifikat eingebunden ist *abarbeiten*
5. Einen signierten Zeitstempel aus der Signatur *abarbeiten*

Dann das *Abarbeiten* über die einzelnen Zertifikate:

1. AuthorityInfoAccess auslesen für Issuer-Zertifikat aus dem Netz und der OCSP URL
2. CRL auslesen. Anzumerken ist, dass nur die erste URL ausgelesen wird. Oft ist auch eine LDAP-URL angegeben welche nicht bearbeitet wird.
3. Überprüfen ob Zertifikat selbst-signiert ist, was heisst, dass es Root ist und nicht weiter traversiert werden muss.

4. Der Issuer wird aus dem Zertifikat-Pool gesucht und mit diesem wieder bei Schritt 1 abgearbeitet.

4.2.2.2 *CertSignatureInformation*

Alle benötigten Informationen werden in die innere Klasse **CertSignatureInformation** von CertInformationCollector gespeichert. Dies aus dem Grund, dass diese Klasse ohne den Collector keinen Sinn ergibt.

Neben den allgemeinen Feldern gibt es drei Felder vom selben Typ, die zum einen die beiden möglichen Zertifikatketten beinhalten und das Dritte beinhaltet die Informationen / Zertifikatkette der TSA. Dies wird damit begründet, dass sämtliche benötigte Informationen auf einmal zurückgegeben werden können.

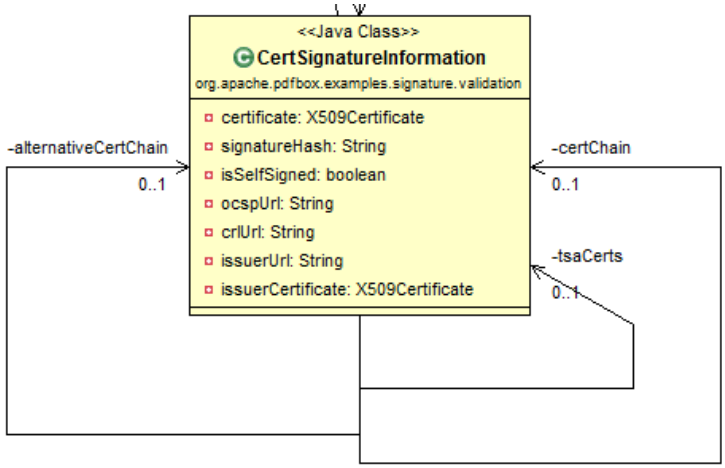
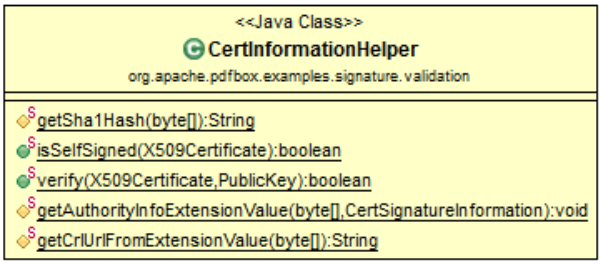


Abbildung 38 Die Klasse CertSignatureInformation ist vor allem Informationsträger über die Signaturen

4.2.2.3 *CertInformationHelper*

Hält allgemein nützliche Methoden für den Collector und andere Anwendungen bereit.
Die Methode **getAuthorityInfoExtensionValue** sucht nach der OCSP-URL und der Issuer-Zertifikat URL. Diese Felder müssen mittels OID gefunden werden.
Die Methode **getCrlUrlFromExtensionValue** extrahiert die erste CRL-URL.
Die Methode **verify** überprüft auf einfache Art ob ein Zertifikat durch den gegebenen public Key signiert wurde.



4.2.3 Validierungsinformationen dem Dokument anfügen

Die Klasse **AddValidationInformation** prüft die Validierungen und bettet diese in das Dokument ein. Hierzu muss die CertSignatureInformation vom Collector durchgegangen werden um sämtliche Zertifikate zu validieren oder einzubetten. Es wird zudem dafür gesorgt, dass Zertifikate nicht doppelt validiert werden.
Um die korrekte DSS-Struktur aufzubauen, muss geprüft werden, ob das Dokument bereits ein DSS beinhaltet, damit dieses korrekt erweitert wird.

4.3 Testing & Codequalität

Um die Funktionalität zu prüfen wurden sehr viele Integrationstests mit verschiedenen Szenarien durchgeführt, welche mit den endgültigen Resultaten in den Ergebnissen dokumentiert sind.

Da in den meisten Funktionen sehr viele Abhängigkeiten zu äusseren Bibliotheken vorhanden sind, ist die Implementierung von Unit-Tests sehr aufwändig.

Um die Weiterentwicklung und -verwendung zu ermöglichen, wurde intensiv in Cleancode investiert.

5 Projektplanung

Zu Beginn des Projektes wurde eine Zeitplanung erstellt, dieser ist hier zur Retrospektive unverändert aufgeführt.

5.1 Projektübersicht

Das Ziel ist das Erstellen einer Erweiterung des vorhanden Apache-Projektes PDFBox. PDFBox ist eine Software-Bibliothek für die Erstellung und Anpassung von PDF Dokumenten. In diesem Projekt soll die Bibliothek so erweitert werden, dass Signaturen auch für LTV korrekt erstellt werden können.

Es gibt dabei diverse Punkte zu beachten. LTV basiert auf einer EU Richtlinie, die auch in der Schweiz übernommen wurde. PAdES – PDF Advanced Electronic Signature Profiles definiert im Profil «PAdES-LTV – PAdES Long Term» die genauen Spezifikationen dieser Signatur-Methode.

Technisch gesehen sieht die Umsetzung die Einbindung der Validationsdaten vor, zu denen die Zertifikat-Kette(n), Einfügung von validierten Zeitstempeln durch eine TSA und Validierungsberichte der Zertifikate. Für die Validierungsberichte der Zertifikate wird nur auf OCSP gesetzt, da diese Zeitgemäss sind.

5.2 Projektorganisation

Das Projekt wurde wegen des vorzeitigen Ausfalls des Partners alleine durchgeführt. Die Aufgabenstellung blieb in seinem Rahmen aber bestehen, da diese schwer unterteilbar ist.

5.3 Zeitplanung

Das Projekt wird im Zeitraum 18. September 2017 bis 22. Dezember 2017 verlaufen.

5.3.1 Phasen / Iterationen

Phase	Iterationen	Start-Datum	End-Datum	Zugeordnete Aufgaben
Inception 18.09.2017-24.09.2017	Inception	18.09.2017	24.09.2017	Kickoff
Elaboration 25.09.2017-15.10.2017	Elaboration_1	25.09.2017	08.10.2017	Projektplan & Risiken
	Elaboration_2	09.10.2017	15.10.2017	Anforderungen und Analyse
Construction 16.10.2017-03.12.2017	Construction_1	16.10.2017	29.10.2017	TimeStamp
	Construction_2	30.10.2017	12.11.2017	OCSP
	Construction_3	13.11.2017	26.11.2017	Testing
	Construction_4	27.11.2017	3.12.2017	Verbesserungen
Transition 04.12.2017-22.12.2017	Transition	04.12.2017	22.12.2017	Dokumentation / Abgabe

5.4 Risikomanagement

Anfangs des Projektes wurden einzelne Risiken aufgespürt und hier dokumentiert.

5.4.1 Zu unklare / offene Spezifikationen

Es ist möglich, dass die LTV-fähigkeit zu schwammig definiert ist. Um dagegen zu halten, sollen alle Features (Timestamp, OCSP) implementiert werden.

5.4.2 Verifikation von LTV-Fähigkeit

Grundsätzlich ist die Verifikation bzw. die Akzeptanz der LTV-Fähigkeit eines PDFs Aufgabe des Empfängers. Hierfür wird zum einen der Validator (http://validator.ch) vom Bund genommen und zum anderen wird der Adobe Reader verwendet. Der Adobe Reader zeigt an, wenn eine Signatur LTV fähig ist. (Aber nicht zu welchem Grad) Da der PDF-Standard auch von Adobe entwickelt wurde, wurde hier die LTV-Fähigkeit von Dokumenten zuerst eingebaut.

5.4.3 PDFBox unterstützt Implementierung nicht

Dies könnte seitens Community oder seitens vorhandenen Code passieren. Die Community sollte unbedingt mit einbezogen werden. Kann aber notfalls umgangen werden, was dazu führt, dass der Code nicht in der offiziellen Library landet.
Seitens Code/Architektur könnte es sein, dass nicht alles unterstützt wird. Das würde heissen, dass Anpassungen für die Unterstützung gemacht werden müssten.

5.5 Infrastruktur

Da die Problemstellung ohne Server klarkommt, wurde sämtliche Analyse und Entwicklung auf einem Client erbracht. Dabei waren diverse Tools im Einsatz unter anderem um PDF-Dokumente und Signaturen zu untersuchen. Entwickelt wurde unter Eclipse, da dies durch PDFBox empfohlen wird.

6 Schlussfolgerung

6.1 Was wurde erreicht

LTV wurde verstanden und konnte implementiert werden.

6.2 Beurteilung Ergebnis

Die LTV ist technisch umgesetzt, doch die Benutzerfreundlichkeit ist noch nicht gegeben, da bisher kaum ein Weg um Adobe Acrobat führte.
Für die gesamte Signatur-Problematik ist die Benutzbarkeit auf einem schweren Stand, da sich wenige User mit Keys herumschlagen möchten.
Für Spezialisten und auch juristische Personen liegt die Anwendung von LTV in naher Zukunft klar da.

6.3 Weiteres Vorgehen

Auf Basis der Resultate können nun Anwendungen der LTV entstehen, um diese für Benutzer einfacher zugänglich zu machen.

Es können diverse Workflows entstehen, die das Signieren und LTV-validieren automatisieren. Insbesondere soll es auch eine Möglichkeit geben, eine Signatur offline zu erstellen, danach die Validierung und Timestamping über das Netz einzubinden.

Eine Möglichkeit wäre die Verifizierung der LTV. Die visuell aufzeigt ob ein signiertes Dokument LTV-fähig ist. Ähnlich dem validator.ch.

Zudem gibt es noch eine Optimierungsmöglichkeit beim Vorgehen, die ein vorhandenes DSS einliest und bereits vorhandene Daten nicht ein zweites mal in das Dokument einbindet.

Für den Fall, dass bei OCSP keine Nonce/Attribute akzeptiert werden, sollte abgeklärt werden ob es eine Alternative Lösung zur CRL gibt. Ansonsten funktioniert die Lösung wie sie ist.

Für PDFBox könnte der DSS in die Bibliothek selber, ähnlich der Signatur hinzugenommen werden. Es ist dabei fraglich wieviel in der Bibliothek selber stecken soll und wieviel in der Anwendung selber

stecken soll.

6.4 Zeitaufwand

Für die Erarbeitung wurden insgesamt 384 Stunden aufgewendet. Davon 35% Dokumentationen und Standards nachlesen, 35% Implementierung und Anpassungen, 28% Dokumentation und 2% Kommunikation.

7 Literaturverzeichnis

7.1 ETSI

ETSI TS 102 778-x
Part 1: PAdES Overview - a framework document for PAdES
http://www.etsi.org/deliver/etsi_ts/102700_102799/10277801/01.01.01_60/ts_10277801v010101p.pdf

Part 2: PAdES Basic - Profile based on ISO 32000-1
http://www.etsi.org/deliver/etsi_ts/102700_102799/10277802/01.02.01_60/ts_10277802v010201p.pdf

Part 3: PAdES Enhanced - PAdES-BES and PAdES-EPES Profiles
http://www.etsi.org/deliver/etsi_ts/102700_102799/10277803/01.02.01_60/ts_10277803v010201p.pdf

Part 4: PAdES Long Term - PAdES-LTV Profile
http://www.etsi.org/deliver/etsi_ts/102700_102799/10277804/01.01.02_60/ts_10277804v010102p.pdf

Part 5: PAdES for XML Content - Profiles for XAdES signatures
http://www.etsi.org/deliver/etsi_ts/102700_102799/10277805/01.01.02_60/ts_10277805v010102p.pdf

Part 6: Visual Representations of Electronic Signatures
http://www.etsi.org/deliver/etsi_ts/102700_102799/10277806/01.01.01_60/ts_10277806v010101p.pdf

7.2 ISO

PDF: ISO 32000-1:2008 - Document management -- Portable document format -- Part 1: PDF 1.7
www.adobe.com/devnet/acrobat/pdfs/PDF32000_2008.pdf

7.3 RFC

Cryptographic Message Syntax (CMS)
https://tools.ietf.org/html/rfc5652

Online Certificate Status Protocol – OCSP
https://tools.ietf.org/html/rfc6960

Time-Stamp Protocol (TSP)
https://tools.ietf.org/html/rfc3161

Certificate Revocation List (CRL) Profile
https://tools.ietf.org/html/rfc5280

7.4 Weiterführende Literatur

ETSI Dokument-Suche
http://www.etsi.org/standards-search

Adobe - Digital Signatures in a PDF
https://www.adobe.com/devnet-docs/acrobatetk/tools/DigSig/Acrobat_DigitalSignatures_in_PDF.pdf

iText 7: Digital Signatures for PDF
https://developers.itextpdf.com/content/itext-7-digital-signatures-pdf
https://pages.itextpdf.com/ebook-digital-signatures-for-pdf.html

ASN.1 Visualizer
http://aaa-sec.com/other/jsasn1/index.html

8 Anhänge

8.1 Persöhnlicher Bericht

Mit dem Novum im Sicherheitsbereich eine Arbeit zu leisten konnte ich einiges für diesen Bereich dazulernen. Die Arbeit statt zu zweit zu erarbeiten erforderte einige Anstrengung und Disziplin, hatte aber den eindeutigen Vorteil, dass nur ich das ganze Thema verstehen musste. Von Vorteil war auch meine Erfahrung im Java-Bereich, was den Gebrauch von PDFBox sehr erleichtert.

Der Einstieg in das ganze Thema um LTV war etwas mühsam, da ich mich wie im Nebel fühlte. Es wurde mit sehr vielen Standards / ISOs und RFCs herumgeworfen. Einige Punkte wurden anfangs missverstanden oder übersehen. Manches wurde sogar gewollt ausgelassen, da ich es für unnötig hielt. Ein Fehler wie sich herausstellte, der aber nach dem Entdecken extrem lehrreich war. Dadurch erlernte ich das Lesen von Dokumenten aus verschiedenen Institutionen und entdeckte sehr ausgeklügelte Zusammenhänge. Beispielsweise mag die Anwendung von ASN.1 anfangs sehr verstörend wirken. Doch die Implementierung von CMS zeigt sich als sehr praktisch und ist bereits seit bald 20 Jahren im Einsatz.

Die Arbeit erforderte ab einem gewissen Wissenslevel nur noch wenig Zusammenarbeit und externe Abklärungen was ich auch meiner Selbständigkeit verdanke. Anderseits war mir dadurch nicht immer klar, wieviel von mir erwartet wurde. Erleichterung gab aber, dass die einzelnen Schritte mit Zuversicht bis hin zur LTV erreicht wurden und die Lösung im PDFBox Projekt akzeptiert wurde.

Die genaue Planung des Projektes zu beginn war schwierig, da einige Details noch unbekannt waren. Deshalb gab es gegen Ende der Arbeit noch einzelne Verbesserungsaufgaben, die aber nicht zu viel Zeit in Anspruch nahmen.

Rundum bin ich glücklich über das Resultat und zähle mich nun zu einem Eingeweihten der LongTerm Validation.

8.2 Abkürzungsverzeichnis

Begriff	Erklärung
LTV	Long Term Validation einer Signatur. In diesem Dokument geht es spezifisch um die Signatur in PDF Dokumenten.
Apache	Kurz für Apache Software Foundation. Organisation zur Förderung von Apache-Softwareprojekten.

TSA	Time Stamping Authority, signiert einen gegebenen Hash und einen zeitnahen Timestamp um den Existenzzeitpunkt des Hashs sicherzustellen.
VRI	Validation-Related Information
ETSI	Europäisches Institut für Telekommunikationsnormen
PAdES	PDF Advanced Electronic Signature, Spezifikationen für PDF-Signaturen, durch ETSI bereitgestellt. Es gibt mehrere Profile der Spezifikationen, die in diesem Dokument tangiert werden.
DSS	Document Signer Store

8.3 Eigenständigkeitserklärung

Ich erkläre hiermit,

- dass ich die vorliegende Arbeit selber und ohne fremde Hilfe durchgeführt habe, ausser derjenigen, welche explizit in der Aufgabenstellung erwähnt ist oder mit dem Betreuer schriftlich vereinbart wurde,
- dass ich sämtliche verwendeten Quellen erwähnt und gemäss gängigen wissenschaftlichen Zitierregeln korrekt angegeben habe.
- dass ich keine durch Copyright geschützten Materialien (z.B. Bilder) in dieser Arbeit in unerlaubter Weise genutzt habe.

Rapperswil, 12. Dezember 2017

Alexis Suter