

Die CCES-Signature-API

- eine offene Programmierschnittstelle für langfristig beweiskräftige elektronische Signaturen

Detlef Hühnlein

secunet Security Networks AG
Sudetenstraße 16, 96247 Michelau
detlef.huehnlein@secunet.com

Abstract: Dieser Beitrag liefert einen Überblick über die CCES-Signature-API – eine im Competence Center Elektronische Signatur (CCES) des Verbandes Organisations- und Informationssysteme e.V. (VOI) spezifizierte Programmierschnittstelle für elektronische Signaturen. Entgegen existierender Standardschnittstellen für kryptographische Operationen berücksichtigt diese API die spezifischen Anforderungen aus dem Dokumentenmanagement und ermöglicht die Unterstützung verschiedener Daten- und Signaturformate, Signaturerstellungseinheiten und Gültigkeitsmodelle sowie die Verwendung von Mehrfach- und Massensignaturen, Zeitstempeln und erweiterte Funktionen zum langfristigen Erhalt des Beweiswertes elektronischer Signaturen.

1 Einleitung

Die Integration der elektronischen Signatur in Applikationssysteme erfolgt zumeist durch Einbindung spezialisierter Bibliotheken, die die notwendige Funktionalität über eine Programmierschnittstelle (Application Programming Interface, API) bereitstellen.

Wie in Abschnitt 2.2 ersichtlich, existieren bereits viele öffentlich zugängliche APIs durch die kryptographische Funktionen in beliebige Anwendungen integriert werden können. Betrachtet man jedoch die typischen Anforderungen an eine API für elektronische Signaturen etwas genauer, so erweisen sich diese Standardschnittstellen oft als wenig geeignet. Müssen verschiedene Daten- und Signaturformate, Signaturerstellungseinheiten, Gültigkeitsmodelle, Mehrfach- und Massensignaturen, Zeitstempel und erweiterte Mechanismen zum langfristigen Erhalt des Beweiswertes qualifizierter elektronischer Signaturen unterstützt werden, so ist dies oft nur mit erheblichem individuellen Entwicklungsaufwand oder der Nutzung proprietärer Funktionsbibliotheken und Programmierschnittstellen möglich. Sofern Bibliotheken verschiedener Anbieter unterstützt werden müssen, steigt der Integrationsaufwand weiter. Deshalb wurde im Competence Center Elektronische Signatur (CCES) des Verbandes Organisations- und Informationssysteme e.V. (VOI) eine herstellerunabhängige Programmierschnittstelle spezifiziert, mit der langfristig beweiskräftige elektronische Signaturen leicht in beliebige Anwendungen, wie zum Beispiel Dokumenten-Management-Systeme (DMS), integriert werden können.

2 Motivation für die Entwicklung der CCES-Signature-API

In diesem Abschnitt ist die grundlegende Motivation erläutert, die zur Spezifikation der CCES-Signatur-API geführt hat. Hierzu skizzieren wir in Abschnitt 2.1 einige beispielhafte Anwendungsfälle für elektronische Signaturen im DMS-Umfeld und leiten daraus in Abschnitt 2.2 die typischen Anforderungen an eine Signatur-API im DMS-Umfeld ab. In Abschnitt 2.3 betrachten wir existierende APIs für kryptographische Operationen vor dem Hintergrund dieser Anforderungen. Die Ergebnisse dieser Analyse finden sich in Abschnitt 2.4.

2.1 Beispielhafte Anwendungsfälle für elektronische Signaturen im DMS-Umfeld

In der DMS-Branche widmet man sich der Erfassung, Verwaltung / Verarbeitung, Bereitstellung / Ausgabe, Speicherung und Archivierung von Informationen zur Unterstützung der Geschäftsprozesse in Unternehmen und Behörden. In diesem Umfeld existieren beispielsweise folgende Anwendungsfälle der elektronischen Signatur:

1. Der *Nachweis der Übereinstimmung eingescannter Belege mit Original* gemäß § 110a Abs. 2 Nr. 1 a) [SGB IV] kann¹ unter Verwendung elektronischer Signaturen geführt werden. Um auch große Mengen schriftlicher Unterlagen einscannen und signieren zu können, muss die Massensignatur berücksichtigt werden (vgl. [HüKn03]).
2. Sollen *Workflow-Prozesse* mit Formerforderniss, z.B. zur Feststellung (§§ 19-21 [SRVwV]) und Zahlungsanordnung (§ 11 [SRVwV]) bei Sozialversicherungsträgern elektronisch abgewickelt werden, so müssen qualifizierte elektronische Signaturen eingesetzt und (parallele und sequentielle) Mehrfachsignaturen unterstützt werden.
3. Die qualifizierte elektronische Signatur an *elektronisch übermittelten Rechnungen* (vgl. § 14 Abs. 3 [UStG] und GDPdU [BMF01]) kann grundsätzlich in einem beliebigen Signaturformat vorliegen. Deshalb sollte ein Archivsystem auch mit beliebigen Signaturformaten, wie z. B. PKCS#7 / CMS, XML-DSig, S/MIME und in PDF integrierte PKCS #7 Signaturen umgehen können.
4. Für die beweiskräftige *elektronische Archivierung* bietet sich der Einsatz vertrauenswürdiger kryptographisch gesicherter Zeitstempel an.
5. Soll die erhöhte Beweiskraft qualifizierter elektronischer Signaturen langfristig erhalten werden, so müssen u.U. *Nachsichtungen* im Sinne von § 17 [SigV] erfolgen. Bei einer großen Menge an Signaturen bietet sich die Nutzung der Evidence Record Syntax (ERS) [BrHu04] oder der Einsatz intervall-qualifizierter Zeitstempel [Hüh04] an.

¹ Bis vor kurzem war der Einsatz qualifizierter elektronischer Signaturen auf Grund von §36 [SRVwV] beim Einscannen zahlungsbegründender Unterlagen zwingend vorgeschrieben (vgl. [HüKn03]).

2.2 Anforderungen an eine Signatur-API im DMS-Umfeld

Eine Signatur-API für die Nutzung im DMS-Umfeld sollte folgende Funktionen bereitstellen:

- *Erstellung und Verifikation elektronischer Signaturen* unter Berücksichtigung folgender Aspekte:
 - Mengengerüste (Einzelsignatur oder Massensignatur)
 - Qualitätsstufen (von der einfachen elektronischen Signatur bis hin zur qualifizierten elektronischen Signatur mit Anbieterakkreditierung)
 - Mehrfachsignaturen (parallel und sequentiell)
 - Personal Security Environments (PSEs) (handschriftlich, Software-PSE, Hardwaretoken)
 - Gültigkeitsmodelle (Kettenmodell und Schalenmodell)
 - Systemarchitekturen (zentrale und dezentrale Signaturerzeugung)
 - Signaturformate
 - Cryptographic Message Syntax (CMS) [RFC3369] bzw. PKCS#7 ([PKCS#7] und [RFC2315])
 - XML Digital Signature [RFC3275]
 - E-Mail-Signaturformate
 - S/MIME [RFC2633]
 - OpenPGP [RFC2440] und PGP/MIME [RFC3156]
 - Dokumentenformate mit eingebetteter Signatur, wie z.B. [AI-PDF] oder [AI-TIFF]
- *Anforderung und Verifikation von Zeitstempeln* über das Time Stamp Protocol (TSP) [RFC3161].
- *Erweiterte Funktionalität zur langfristigen Archivierung* durch Unterstützung der Archive-Time-Stamp Syntax (ATS) [BGPT03], bzw. der darauf aufbauend in [BrHu04] spezifizierten Evidence Record Syntax (ERS).
- *Weitere kryptographische Operationen*, wie z.B. die Verschlüsselung von Daten mit PKCS#7 / CMS, S/MIME und PGP.

2.3 Existierende APIs für kryptographische Operationen

In diesem Abschnitt knüpfen wir an die Vorarbeiten in [BaSch00] an und liefern einen erweiterten und aktualisierten Überblick über bereits existierende Programmierschnittstellen mit Bezug zur elektronischen Signatur und vergleichen diese mit den oben skizzierten Anforderungen.

2.3.1 Acrobat Digital Signature API

Bei der Acrobat Digital Signature API [AI-API-99] handelt es sich um eine Schnittstelle zwischen der Acrobat-Applikation und einem Acrobat-spezifischen Signatur-PlugIn. Der Fokus der vorgesehenen Funktionalität liegt im Integritätsschutz (ausgewählter Teile) einer PDF-Datei durch Hash- und Signaturmechanismen.

Auch wenn bei der aktuellen Spezifikation dieser Schnittstelle [AI-API-03] mit dem „PubSec Layer“ eine auf den Einsatz von Public-Key-Mechanismen zugeschnittene Schnittschicht existiert, so fehlen dennoch weite Teile der benötigten Funktionalität (z.B. Zeitstempel und verschiedener Signatur- und Dokumentenformate).

2.3.2 Common Data Security Architecture / Common Security Services Manager (CDSA / CSSM)

Die CDSA [X/O-CDSA] wurde ursprünglich von Intel entwickelt und von der Open Group als umfassende Sicherheitsarchitektur spezifiziert. Neben den über die CSSM-API in einheitlicher Art und Weise nach außen zur Verfügung gestellten Sicherheitsdiensten, können Module mit zusätzlichen Diensten spezifiziert und angebunden werden. Die standardmäßig vorgesehene Funktionalität umfasst beispielsweise auch das Erstellen und Prüfen elektronischer Signaturen in Low-Level-Formaten wie PKCS #1. Die Unterstützung verschiedener Signaturformate müsste durch das (u. U. gleichzeitige) Laden von mehreren Cryptographic Service Providers, oder aber über die Definition eines mächtigeren Erweiterungsmoduls, realisiert werden. Die Unterstützung von Zeitstempeln und weiteren Mechanismen zur langfristigen Archivierung müsste in einem noch zu spezifizierenden Erweiterungsmodul realisiert werden.

Während es durch die Erweiterbarkeit der CDSA und CSSM also generell möglich wäre, ein maßgeschneidertes Erweiterungsmodul und eine zugehörige API für die Integration der elektronischen Signatur in DMS-Anwendungen zu spezifizieren und als zusätzliches Modul in einer CDSA-Implementierung zu nutzen, so erscheint eine zwangsweise Kopplung der CCES-Signature-API mit CDSA/CSSM eher hinderlich. Dies gilt umso mehr, da CDSA/CSSM derzeit noch immer kaum² verbreitet zu sein scheint.

2.3.3 Generic Cryptographic Service API (GCS-API)

Die GCS-API [X/O-GCS] ist ein lediglich als vorläufige Spezifikation der Open Group verfügbarer Standardisierungsansatz, durch den auf verschiedene kryptographische Algorithmen über generische Aufrufe und Algorithmen-spezifische Templates zugegriffen werden kann. Da es sich hierbei eher um eine Low-Level-API handelt, die zudem wichtige Funktionen, wie z.B. Zeitstempel, vermissen lässt, ist die GCS-API als Basis für die Nutzung im DMS-Umfeld wenig geeignet.

2.3.4 Generic Security Services API (GSS-API)

Bei der in [RFC2078] spezifizierten GSS-API handelt es sich um eine generische Schnittstelle zur sicheren Client-Server-Kommunikation, die beispielsweise im SAP R/3-Umfeld zum Einsatz kommt.

² Neben der mittlerweile als Open Source verfügbaren Referenz-Implementierung von Intel (<http://sourceforge.net/projects/cdsa/>) und der CDSA-Realisierung für Mac/OS von Apple (http://developer.apple.com/documentation/Security/Conceptual/Security_Overview/Architecture/chapter_2_section_3.html) ist keine weitere CDSA-Implementierung bekannt.

Da der Zweck der GSS-API der Schutz von Client-Server-Kommunikationsverbindungen ist, ist sie für Zwecke der elektronischen Signatur generell ungeeignet.

2.3.5 GSS-Independent Data Unit Protection (GSS-IDUP)

Der GSS-IDUP-Standard [RFC2479] ist das in der Praxis vergleichsweise wenig verbreitete, dokumentenbasierte Pendant zur GSS-API. Hierin sind generische Sicherheitsmechanismen für Dokumente definiert, mit denen auch elektronische Signaturen, beispielsweise im PEM-Format [RFC1421], realisiert werden könnten. Leider existieren, anders als bei der in der Praxis eingesetzten GSS-API, noch keinerlei C- oder Java-Bindings im Stile von [RFC2744] bzw. [RFC2853]. Da die in [RFC2479] spezifizierte Funktionalität selbst noch um viele Aspekte ergänzt werden müsste, scheint die Verwendung von GSS-IDUP für die Integration der elektronischen Signatur im DMS-Umfeld wenig Vorteile zu bieten.

2.3.6 Java Cryptographic Architecture (JCA) und Extension (JCE)

Die Java Cryptographic Architecture [JCA] in Verbindung mit der Java Cryptography Extension [JCE], die konkrete kryptographische Mechanismen enthält und aus Gründen der Exporterleichterung von der JCA separiert ist, bildet den weithin akzeptierten Standard für die Unterstützung kryptographischer Primitive in Java. Allerdings handelt es sich hierbei um eine Low-Level-API, die nicht als Ersatz, sondern vielmehr als Basis zur Realisierung eines Java-basierten Providers, für die CCES-Signature-API zu sehen ist.

2.3.7 Java XML Digital Signature API

Bei der Java XML Digital Signature API [JXS-API] handelt es sich um eine Java-basierte Schnittstelle zur Realisierung von XML-Signaturen gemäß [RFC3275]. Hierbei können die in [RFC3275] definierten Signatur-Objekte in einem XML-Dokument angelegt, näher spezifiziert und schließlich signiert und geprüft werden. Während der generelle Charakter dieser Schnittstelle dem der anvisierten CCES-Signature-API ähnelt, so deckt die Java-basierte Schnittstelle [JXS-API] unglücklicherweise nur einen Ausschnitt der benötigten Funktionalität ab. Eine Unterstützung von Zeitstempeln und verschiedenen Signaturformaten ist nicht vorgesehen.

2.3.8 Microsoft CryptoAPI

Die Microsoft CryptoAPI [MS-CAPI] stellt Windows-basierten Applikationen verschiedene kryptographische Mechanismen, wie z.B. die Erzeugung und Verifikation von Signaturen im PKCS #7-Format oder die Prüfung eines Zertifikatspfades nach dem Schalenmodell (bezüglich des Prüfungszeitpunktes³) zur Verfügung.

³ Da für die Gültigkeit von qualifizierten Signaturen der Zeitpunkt der Signaturerstellung maßgeblich ist, führt die naive Verwendung der Microsoft CryptoAPI im Kontext qualifizierter elektronischer Signaturen zu Problemen.

Die Implementierung der kryptographischen Mechanismen selbst erfolgt durch Cryptographic Service Provider. Auch hier kommt der generelle Charakter dieser Schnittstelle dem der anvisierten CCES-Signature-API nahe. Leider wird auch hier lediglich ein Ausschnitt der benötigten Funktionalität abgedeckt. Es fehlt die Unterstützung von Zeitstempeln und verschiedenen Signaturformaten.

2.3.9 Open Card Framework (OCF)

Das Open Card Framework [OCF] definiert eine Schnittstelle zur Java-nativen Ansteuerung von Chipkarten. Durch den Low-Level-Charakter ist diese Schnittstelle als Basis für die Spezifikation der CCES-Signature-API generell ungeeignet.

2.3.10 PC/SC und CT-API

Bei der PC/SC-Schnittstelle [PC/SC] handelt es sich um eine Schnittstelle zwischen – ursprünglich Windows-basierten – PCs und Chipkartenlesern. Ähnliches leistet die CT-API [CT-API] auf verschiedenen weiteren Plattformen. Deshalb sind diese Schnittstellen allein zur Integration der elektronischen Signatur in Anwendungssystemen nicht zu gebrauchen.

2.2.11 PKCS#11

Die Cryptographic Token Interface (CrypTokI) Schnittstelle [PKCS#11] definiert eine Low-Level-Schnittstelle zwischen Anwendungssystemen und kryptographischen Token, wie z.B. Chipkarten oder Hardware-Sicherheitsmodulen. Diese Schnittstelle ist im Rahmen von [ISIS-MTT] (Part 7 – Cryptographic Token Interface) näher profiliert und vergleichsweise weit verbreitet. Da der Funktionsumfang aber auf die Kommunikation mit kryptographischen Token fokussiert ist, ist die alleinige Verwendung dieser Schnittstelle zur Integration der elektronischen Signatur nicht zu empfehlen.

2.3.12 SAP Secure Store and Forward (SSF) API

Mit der SSF-API von SAP [SAP-SSF] wurde eine High-Level-API spezifiziert, mit der Daten im PKCS #7-Format signiert und verschlüsselt werden können. Auch hier entspricht der generelle Charakter dieser Schnittstelle dem der anvisierten CCES-Signature-API. Leider wird auch hier lediglich ein Ausschnitt der benötigten Funktionalität abgedeckt. Es fehlt an der Unterstützung von Zeitstempeln und verschiedenen Signaturformaten, so dass in der Praxis oft über proprietäre Schnittstellen auf zusätzliche Funktionen zugegriffen werden muss.

2.3.13 Signaturbündnis-API

Im Rahmen des Signaturbündnisses wurde kürzlich eine Programmierschnittstelle [SigAll-API] entwickelt, die es erlaubt in einheitlicher Art und Weise auf Signaturkarten verschiedener Anbieter zuzugreifen. Somit besitzt diese Schnittstelle einen ähnlichen Charakter wie PKCS#11 und ist daher nicht als Ersatz, sondern vielmehr als Ergänzung zur CCES-Signatur-API zu verstehen. Siehe auch [Hil05].

2.3.14 Simple Cryptographic Program Interface (Crypto API)

Mit dem Simple Cryptographic Program Interface [RFC2628] wurde eine einfache Programmierschnittstelle spezifiziert, die die notwendige kryptographische Basis-Funktionalität zum Aufbau von Virtuellen Privaten Netzen mittels IPSEC oder SSL/TLS bereitstellt. Deshalb sieht diese Schnittstelle derzeit insbesondere Funktionen zur Authentisierung, Ver- und Entschlüsselung sowie Komprimierung vor. Die Nutzung der Schnittstelle zur Signatur von Dokumenten wäre lediglich unter Verwendung von Low-Level-Formaten, wie z.B. PKCS #1 [RFC3447], möglich. Die notwendige High-Level-Funktionalität zur Behandlung von Zertifikaten und Zeitstempeln fehlt leider gänzlich.

2.3.15 US Government Smart Card Interoperability Specification (GSC-IS)

Die vom NIST standardisierte Government Smart Card Interoperability Specification (GSC-IS) der US-Verwaltung [US-GSC-IS] definiert eine generische Schnittstelle für den Zugriff auf verschiedenartige Chipkarten. Neben Chipkarten mit ISO7816-basiertem Dateisystem werden beispielsweise auch Java-Karten über das einheitliche „Virtual Card Edge Interface“ angesprochen. Die generelle Zielsetzung dieser Schnittstelle ist somit im Wesentlichen vergleichbar mit der von Chipkartenspezifikationen im Stile von [TTT-GOIC] und [HPC-Spec], und damit zur alleinigen Nutzung zur Integration der elektronischen Signatur nicht geeignet.

2.4 Zusammenfassende Bewertung der Eignung existierender APIs

Betrachtet man die Komponenten beim Signaturanwender, so ist im Wesentlichen zwischen der Signaturerstellungseinheit (vgl. §2 Nr. 10 SigG), der Signaturanwendungskomponente (vgl. §2 Nr. 11 SigG) und dem eigentlichen Anwendungssystem zu unterscheiden. Die Taxonomie der hier betrachteten Schnittstellen auf Basis dieses einfachen Modells liefert das folgende Bild:

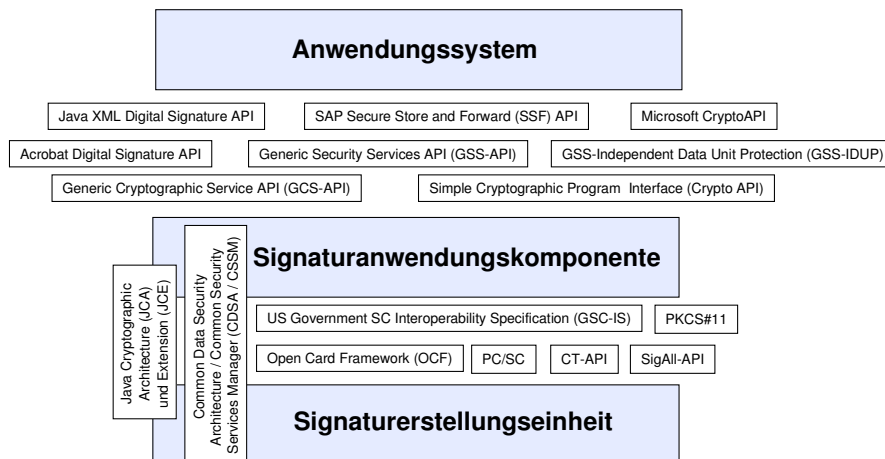


Abbildung 1: Existierende APIs für kryptographische Operationen im Überblick

Betrachtet man nun die oben aufgeführten APIs, dann fällt auf, dass sich einige Standard-Schnittstellen generell nicht als Basis für die CCES-Signature-API eignen, da sie auf einer zu tiefen, Chipkarten-nahen Schicht operieren und dem Anwendungsentwickler noch zu viele Aufgaben bei der Integration der elektronischen Signatur überlassen würden. Aus diesem Grund sind die Standards [OCF], [PC/SC], [CT-API], [PKCS#11], [US-GSC-IS] und [SigAll-API] für eine alleinige Nutzung generell wenig geeignet.

Auf der anderen Seite decken viele Standards, deren genereller Charakter dem der anvisierten CCES-Signature-API entspricht, lediglich eine mehr oder weniger große Untermenge der benötigten Funktionalität ab. Dies gilt beispielsweise für [AI-API-03], [X/O-GCS], [RFC2479], [JCA]/[JCE], [JXS-API], [MS-CAPI], [SAP-SSF] und [RFC2628]. Insbesondere fehlen hier bei all diesen Schnittstellen die für die DMS-Branche wichtigen Funktionen für Zeitstempel und zur langfristigen Archivierung. Deshalb hat man sich im CCES dazu entschlossen, mit der CCES-Signature-API eine neue Schnittstelle zu entwickeln, die die speziellen Anforderungen der DMS-Anbieter berücksichtigt.

3 Die CCES-Signature-API

Dieser Abschnitt liefert einen kompakten Überblick über die CCES-Signature-API, wobei wir uns hier mit einer groben Skizze der wesentlichen Funktionen und Designentscheidungen begnügen und für die detaillierte Spezifikation auf [CCES-API] verweisen.

3.1 Grundlegende Eigenschaften und Datenstrukturen

Die CCES-Signature-API sieht vor, dass mehrere Instanzen einer CCES-Signature-Library parallel verwendet werden können. Der jeweiligen Instanz ist ein bestimmtes Personal-Security-Environment zugeordnet, das bei der Instanziierung der Bibliothek übergeben wird. Zur Signaturerstellung können lokal angeschlossene Chipkarten, Software-PSEs oder Erfassungsgeräte für handschriftliche Unterschriften sowie entfernte Signaturserver über Client-Server-Protokolle im Stile von [RFC3161] verwendet werden.

Alle Funktionsaufrufe haben folgenden generellen Aufbau:

```
CCES_Funktion ( Parameter-1:CCES_Datentyp-1,
                Parameter-2:CCES_Datentyp-2,
                ... ):CCES_Fehlercode
```

Neben Datentypen für administrative Zwecke, wie z.B. zur Adressierung von Bibliotheksinstanzen (CCES_InstanceHandle) und PSEs (CCES_QueryPSEHandle), sind insbesondere die folgenden Typen erwähnenswert:

- Der generische Typ CCES_DataContainer

wird unter anderem zur Übergabe von zu signierenden Daten, Zertifikaten und Signaturen verwendet.

Beispielsweise können zu signierende Daten nicht nur als OctetString, sondern auch in Form von Listen von URLs (z.B. lokalen Dateinamen) oder bereits in gehashter Form (als einzelner Hashwert oder als komplexer Hashbaum) übergeben werden. Neben unstrukturierten Binärdaten werden folgende Datenformate unterstützt:

- MIME gemäß [RFC2045]
 - CMS gemäß [RFC3369]
 - S/MIME gemäß [RFC2633]
 - OpenPGP gemäß [RFC2440]
 - XML gemäß [XML]
 - PDF gemäß [AI-PDF]
- Datentypen für die Behandlung von Prüfungsergebnissen,
wie z.B. `CCES_VerificationResultArray`,
`CCES_CheckSignature` und `CCES_CheckCertificatePath`.
- Datentypen für die Spezifikation von Aufrufoptionen,
wie z.B. `CCES_SignOptions`, `CCES_HashOptions` und
`CCES_VerifyOptions`.

3.2 Funktionen zur Erstellung und Prüfung von elektronischen Signaturen

Für die Erstellung und Prüfung von elektronischen Signaturen sieht die CCES-Signature-API folgende Funktionen vor:

- `Sign`
Diese Funktion wird zur Erstellung elektronischer Signaturen verwendet. Neben den zu signierenden Daten in den oben angegebenen Formaten können auch bereits existierende Signaturen und Attributzertifikate übergeben werden, die in die zu erstellende Signatur eingebunden werden sollen. Abhängig vom übergebenen Datenformat und den `CCES_SignOptions` können Signaturen in folgenden Formaten erzeugt werden:
 - CMS gemäß [RFC3369] (Enveloping und Detached)
 - S/MIME gemäß [RFC2633] (ClearSigned und OpaqueSigned)
 - OpenPGP gemäß [RFC2440]
 - XML-DSig gemäß [RFC3275] (Enveloping, Enveloped und Detached)

Außerdem können beliebige über Objekt-IDs referenzierbare Low-Level-Signaturformate, wie z.B. PKCS #1 [RFC3447] verwendet und PKCS#7 - Signaturen in PDF-Dateien integriert werden. Für die Massensignatur kann eine Liste von URLs bzw. Dateinamen übergeben werden.

- Hash
Durch diese Funktion, die zur Erzeugung von einzelnen Hashwerten oder komplexen Hashbäumen verwendet werden kann, ist der Vorgang des Hashens vom eigentlichen Signiervorgang separiert. Somit ist beispielsweise die effiziente Übermittlung von Hashwerten an einen entfernt operierenden Signatur- oder Zeitstempeldienst möglich. Durch Übergabe der `CCES_HashOptions` kann die zu verwendende Hashfunktion gewählt werden. Wird eine Liste von Dateien übergeben, so kann entweder eine Liste von einzelnen Hashwerten oder aber ein einzelner Hashbaum, der alle Werte enthält, zurück geliefert werden.
- CountSignatures
Durch diese Funktion kann die Anzahl und Struktur der in einem Dokument befindlichen Signaturen ermittelt werden.
- Verify
Mit der `Verify`-Funktion können einzeln spezifizierte Signaturen oder alle existierenden Signaturen für ein, bzw. in einem Dokument überprüft werden. In Abhängigkeit von den übergebenen `CCES_VerifyOptions` kann angegeben werden, ob eine Prüfung des Zertifikatspfades stattfinden soll oder nicht. Diese Prüfung kann lokal mit Sperrlisten oder online über OCSP erfolgen. Außerdem kann die aktuelle Eignung der eingesetzten kryptographischen Signatur- und Hashalgorithmen geprüft werden.
- ValidateCertificate
Die `ValidateCertificate`-Funktion erlaubt die separate Prüfung eines Zertifikatspfades. Wichtig ist, dass hierbei der für die Prüfung betrachtete Referenzzeitpunkt explizit angegeben werden kann, so dass die Realisierung verschiedener Gültigkeitsmodelle, z.B. gemäß X.509 oder SigG, möglich ist.

3.3 Funktionen zum langfristigen Erhalt des Beweiswertes elektronischer Signaturen

Für den langfristigen Erhalt des Beweiswertes elektronischer Signaturen sieht die CCES-Signature-API die Verwendung von Zeitstempeln sowie erweiterte Funktionen zur Erzeugung, Erneuerung und Prüfung so genannter Evidence Records vor (vgl. [BrHu04]). Insgesamt stehen folgende Funktionen zur Verfügung:

- GetTimestamp
Mit dieser Funktion können Zeitstempel gemäß [RFC3161] oder erweiterte Archiv-Zeitstempel gemäß [BrHu04] erzeugt werden. Die Adressierung des Zeitstempeldienstes erfolgt durch Übergabe einer URL. Die Zeitstempelanfrage kann optional mit einer Signatur versehen werden.

- `VerifyTimestamp`

Durch diese Funktion kann – analog zur Prüfung einer elektronischen Signatur mit `Verify` – die Gültigkeit von Zeitstempeln überprüft werden.

- `CreateEvidenceRecord`

Mit dieser Funktion kann ein Evidence Record gemäß [BrHu04] erzeugt werden. Die Übergabe der Daten erfolgt im Klartext als eine Folge einzelner Hashwerte oder in Form eines Hashbaumes.

- `RenewEvidenceRecord`

Diese Funktion erlaubt die Erneuerung bereits existierender Evidence Records, wobei zwischen der einfachen Übersignatur und der kompletten Rekonstruktion des Hashbaumes gewählt werden kann.

- `VerifyEvidenceRecord`

Mit dieser Funktion kann – analog zur Prüfung einer elektronischen Signatur oder eines Zeitstempels – die Gültigkeit eines Evidence Records überprüft werden.

3.4 Weitere Funktionen

Neben den oben erläuterten Funktionen für langfristig beweiskräftige elektronische Signaturen bietet die CCES-Signature-API auch Funktionen zur Ver- und Entschlüsselung von Daten in den Formaten CMS [RFC3369], S/MIME [RFC2633] und OpenPGP [RFC2440].

4 Zusammenfassung und Ausblick

Dieser Beitrag lieferte einen aktuellen Überblick über existierende APIs für kryptographische Operationen und diskutierte deren Eignung für die alleinige Verwendung zur Integration der elektronischen Signatur in Applikationssysteme.

Da sich die existierenden APIs bei dieser Analyse als wenig geeignet erwiesen, wurde im CCES des VOI e.V. mit der CCES-Signature-API [CCES-API] eine offene Programmierschnittstelle für langfristig beweiskräftige elektronische Signaturen entwickelt, durch die die Integration der elektronischen Signatur in Dokumentenmanagement- und Archivsysteme leicht möglich sein sollte.

Da bereits mehrere Anbieter von Dokumentenmanagement- und Signatursystemen eine Unterstützung der Schnittstelle angekündigt haben, darf man bald mit den ersten praktischen Umsetzungen rechnen. Außerdem berät man derzeit im CCES über mögliche Schritte zur weiteren Standardisierung und Verbreitung dieser Schnittstelle.

Literaturverzeichnis

- [AI-API-99] Adobe Inc.: *Acrobat Digital Signature API*, Adobe Developer Technologies – Technical Note #5192, revised November 10th, 1999, via <http://partners.adobe.com/asn/acrobat/docs/digsig.pdf>
- [AI-API-03] Adobe Inc.: *Acrobat Digital Signature API*, Adobe Developer Technologies – Technical Note #5192, Version: Acrobat 6.0, May 2003, via http://partners.adobe.com/asn/acrobat/sdk/reg/Documentation/Extended_API_For_Plugins/DigitalSignatureAPIRef.pdf
- [AI-PDF] Adobe Inc.: *Portable Document Format Reference Manual*, Version 1.3-1.5, via <http://partners.adobe.com/asn/tech/pdf/specifications.jsp>
- [AI-TIFF] Adobe Developer Association: *TIFF Specification*, Revision 6.0, June 3rd, 1992, via <http://partners.adobe.com/asn/developer/pdfs/tn/TIFF6.pdf>
- [BaSch00] Bartosch, M., Schneider, J.: *Nutzen und Grenzen von Kryptographie-Standards und ihrer APIs*, Tagungsband „Systemsicherheit“, DuD Fachbeiträge, Vieweg, 2000, S. 206-226
- [BMF01] Bundesfinanzministerium: *BMF-Schreiben vom 16.07.01- IV D 2 -S 0316 - 0136/01*, Grundsätze zum Datenzugriff und zur Prüfbarkeit digitaler Unterlagen (GDPdU), via <http://www.bundesfinanzministerium.de/Anlage8440/BMF-Schreiben-vom-16.07.01.pdf>
- [BGPT03] Brandner, R.; Gondrom, T., Pordesch, U.; Tielemann, M.: *Archive Time-Stamps Syntax (ATS)*, Internet-Draft, July 2003 via <http://ltans.edelweb.fr/draft-brandner-et-al-ats-00.txt>
- [BrHu04] Brandner, R.; Hunter, B.: *Evidence Record Syntax*, Internet-Draft, Juli 2004, via <http://ltans.edelweb.fr/draft-ietf-ltans-ers-01.txt>
- [CCES-API] Competence Center Elektronische Signatur im VOI e.V.: *CCES-Signature-API*, Version 1.0 vom 21.01.2005, via <http://www.voi.de>
- [CT-API] Attrott, J.; Eckstein, L.; Kowalski, B.; Moos, R.; Reimer, H.; Struif, B.: *CT-API Anwendungsunabhängiges CardTerminal-Application Programming Interface für Chipkartenanwendungen*, Version 1.1.1, via <https://www.secure.trusted-site.de/Download/CTAPI/CTAPI111.pdf>
- [Hil05] Hillen, D.: *SigBü-API: Einheitliche kartenunabhängige API für die Zugriffe auf die Signaturkarten des Signaturbündnisses*, in diesem Tagungsband
- [HPC-Spec] Struif, B.: *German Health Professional Card and Security Module Card Specification – Pharmacist & Physician*, Version 2.0 vom 31.7.2003, via <http://www.aekwl.de/public/aktuelles/download/pdf/hpc20.pdf>
- [Hüh04] Hühnlein, D.: *Intervall-qualifizierte Zeitstempel*, Tagungsband „Elektronische Geschäftsprozesse“, IT-Verlag, ISBN 3-00-014186-3, 2004, SS. 431-445
- [HüKn03] Hühnlein, D.; Knosowski, Y.: *Aspekte der Massensignatur*, Tagungsband DACH Security 2003, IT-Verlag, 2003, ISBN 3-00-010941-2, SS. 293-307
- [ISIS-MTT] TeleTrusT e.V.: *ISIS-MTT-Spezifikation*, Version 1.1, März 2004, via <http://www.isis-mtt.de/>
- [JCA] Sun Inc.: *Java Cryptographic Architecture (JCA)*, via <http://www.javasoft.com/products/jdk/1.2/docs/guide/security/CryptoSpec.html>
- [JCE] Sun Inc.: *Java Cryptography Extension (JCE)*, via <http://www.javasoft.com/products/jce>
- [JXS-API] Sun Inc.: *XML Digital Signature APIs*, JSR-000105, via <http://jcp.org/aboutJava/communityprocess/review/jsr105/index.html>
- [MS-CAPI] Microsoft Inc.: *Cryptography Reference (Microsoft CryptoAPI)*, Platform SDK: Security, via http://msdn.microsoft.com/library/en-us/security/security/cryptography_reference.asp

- [OCF] Open Card Consortium, *OpenCard Framework V 1.2*, via <http://www.opencard.org/docs/1.2/index.html>
- [PC/SC] PC/SC Workgroup, *PC/SC Workgroup Specifications 1.0*, via <http://pcscworkgroup.com>
- [PKCS#7] RSA Labs: *PKCS #7 Cryptographic Message Syntax Standard*, via <http://www.rsalabs.com/pkcs/pkcs-ia7/index.html> (siehe auch [RFC2315])
- [PKCS#11] RSA Labs: *PKCS#11: Cryptographic Token Interface Standard*, Version 2.11, via <http://www.rsasecurity.com/rsalabs/pkcs/pkcs-11/index.html>
- [RFC1421] Linn, J.: *Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures*, RFC 1421, 1993, via <http://www.ietf.org>
- [RFC2045] Freed, N.; Borenstein, N.: *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*, RFC2045, 1996, via <http://www.ietf.org>
- [RFC2078] Linn, J.: *Generic Security Services Application Programming Interface*, Version 2 (GSS-API v2), RFC2078, 1997, via <http://www.ietf.org>
- [RFC2315] Kaliski, B.: *PKCS #7: Cryptographic Message Syntax*, Version 1.5, RFC 2315, 1998, via <http://www.ietf.org>
- [RFC2440] Callas, J.; Donnerhacke, L.; Finney, H.; Thayer, R.: *OpenPGP Message Format*, RFC 2440, 1998, via <http://www.ietf.org>
- [RFC2479] Adams, C.: *Independent Data Unit Protection – Generic Security Service Application Program Interface (IDUP-GSS-API) v2*, RFC 2479, 1998, via <http://www.ietf.org>
- [RFC2628] Smyslov, V.: *Simple Cryptographic Program Interface (Crypto API)*, RFC 2628, via <http://www.ietf.org>
- [RFC2630] Housley, R.: *Cryptographic Message Syntax (CMS)*, RFC 2630, via <http://www.ietf.org>
- [RFC2633] Ramsdell, B.: *S/MIME Version 3 Message Specification*, RFC 2633, via <http://www.ietf.org>
- [RFC2744] Wray, J.: *Generic Security Service API Version 2 : C-bindings*, RFC2744, 2000, via <http://www.ietf.org>
- [RFC2853] J. Kabat, J.; Upadhyay, M.: *Generic Security Service API Version 2 : Java Bindings*, 2000, via <http://www.ietf.org>
- [RFC3156] Elkins, M., Del Torto, D., Levien, R., Roessler, T.: *MIME Security with OpenPGP*, 2001, via <http://www.ietf.org>
- [RFC3161] Adams, C.; Cain, P.; Pinkas, D.: *Internet X.509 Public Key Infrastructure – Time Stamp Protocol (TSP)*, RFC 3161, via <http://www.ietf.org>
- [RFC3275] D. Eastlake, D.; Reagle, J.; Solo, D.: *(Extensible Markup Language) XML-Signature Syntax and Processing*, RFC 3275, via <http://www.ietf.org>
- [RFC3280] R. Housley, W. Polk, W. Ford, D. Solo: *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, RFC3280, via <http://www.ietf.org>
- [RFC3369] Housley, R.: *Cryptographic Message Syntax (CMS)*, RFC 3369, ersetzt [RFC2630], via <http://www.ietf.org>
- [RFC3447] J. Jonsson, J.; Kaliski, B.: *PKCS #1 – RSA Cryptography Specifications Version 2.1*, 2003, via <http://www.ietf.org>
- [SAP-SSF] SAP AG: *Secure Store and Forward (SSF) API Specification*, via http://www.sap.com/partners/icc/scenarios/pdf/bc_ssf_api.pdf
- [SGB IV] *Sozialgesetzbuch - Viertes Buch (IV) - Gemeinsame Vorschriften für die Sozialversicherung* - (Artikel I des Gesetzes vom 23. Dezember 1976, BGBl. I S. 3845), via http://bundesrecht.juris.de/bundesrecht/sgb_4/index.html
- [SigAll-API] Signaturlbündnis / SRC GmbH: *SigAll-API – Specification of the Application Programming Interface to the Signature Card*, Version 1.0

- [SigV] *Verordnung zur elektronischen Signatur*, vom 16.11.2001 BGBl. 2001 Teil I Nr. 59, S. 3074 ff), via <http://www.iid.de/iukdg/gesetz/SigV161101.pdf>
- [SRVwV] *Allgemeine Verwaltungsvorschrift über das Rechnungswesen in der Sozialversicherung, (SRVwV) vom 15. Juli 1999*, zuletzt geändert durch die Zweite Allgemeine Verwaltungsvorschrift zur Änderung der SRVwV vom 17.12.2004 (BAnz. Nr. 243, Jg. 56 vom 22.12.2004), via <http://www.hvbg.de/d/revision/gruen/kapitel/kap32/>
- [TTT-GOIC] TeleTrusT e.V.: *German Office Identity Card – Elektronischer Dienstausweis*, Version 1.0, 06.07.2000, via http://www.teletrust.de/dokumente/oic_1-0.pdf
- [US-GSC-IS] NIST: *Government Smart Card Interoperability Specification*, Version 2.1., via <http://csrc.nist.gov/publications/nistir/nistir-6887.pdf>
- [UStG] *Umsatzsteuergesetz*, Neugefasst durch Bekanntmachung vom 09.06.1999 BGBl. Teil I Seite 1270 ff, zuletzt geändert durch Änderung durch Art. 5 G v. 9.12.2004 I 3310, via http://bundesrecht.juris.de/bundesrecht/ustg_1980/gesamt.pdf
- [XML] T. Bray, E. Maler, J. Paoli, C. M. Sperberg-McQueen: *Extensible Markup Language (XML) 1.0 (Second Edition)*, W3C Recommendation, 2000, via <http://www.w3.org/TR/2000/REC-xml-20001006>
- [X/O-CDSA] X/Open: *Common Security: CDSA and CSSM*, Version 2.3, Technical Standard, May 2000, ISBN: 1-85912-202-7, via <http://www.opengroup.org/publications/catalog/c914.htm>
- [X/O-GCS] X/Open: *Generic Cryptographic Service API (GCS-API) Base*, Preliminary Specification, June 1996, ISBN: 1-85912-195-0, via <http://www.opengroup.org/publications/catalog/p442.htm>