e-Signatur in Österreich

„Wo stehen wir, wohin geht die Reise?“

zur Lage der Nation – eine Bestandsaufnahme zur Akzeptanzverbesserung der elektronischen Signatur

Studie im Rahmen der netidee 2010

mit freundlicher Unterstützung der Internet Privatstiftung Austria (IPA)

DDipl.-Ing. Mag. Gernot Schmied
Ing. Wolfgang Fabics

Inhaltsverzeichnis

Vorwort

Erklärtes Motiv dieser Studie ist die Förderung der barrierefreien und intuitiven Anwendbarkeit der digitalen Signatur, der Abdeckung sinnvoller und alltagstauglicher *Use Cases* und die Verbesserung der *User Experience* für den interessierten Laien.

Unserer Überzeugung nach müssen Installation und Verwendung digitaler Signaturverfahren für den durchschnittlichen PC-Benutzer ohne die Lektüre von Handbüchern bewältigbar, d.h. aus dem gewohnten Arbeitskontext heraus alleine durch Ausprobieren erfassbar, überschaubar und erlernbar sein (Benutzerfreundlichkeit, intuitive Gestaltung, *Complexity Hiding*, selbsterklärende Anwendungen, verständliche Begriffe). Erst dann, wenn das Signieren digitaler Dokumente *de facto* so einfach ist wie einen Kugelschreiber zu zücken, ist die Basis für den flächendeckenden elektronischen Ersatz der „eigenhändigen digitalen Unterschrift“ gegeben.

Wesentliches Ziel der Studie ist es, durch kritische Betrachtung die Diskussion um die Zukunft der digitalen Signaturen in Österreich erneut anzufachen und konstruktiv zu beleben. Zielpublikum sind somit alle interessierten Kreise bzw. Stakeholder. Insbesondere interessiert uns natürlich das Feedback, ob wir mit unseren Schlussfolgerungen die Nöte sowohl der Anwender als auch der Entwickler im freien Markt richtig zusammengefasst haben.

Besonders bedanken möchten wir uns bei

der ***Internet-Privatstiftung Austria (IPA)***, ohne deren Förderung diese umfassende und sehr zeitaufwändige Studie nicht zustande gekommen wäre, insbesondere bei Herrn Dr. Rupert Nagler für die freundliche Betreuung und Koordination,

Herrn ***Dipl.-Ing. Herbert Leitold*** (EGIZ) für die umfangreiche fachliche Unterstützung und wertvolle Einblicke in Ziele und Sachzwänge der BKU-Entwicklung,

der ***IT Solution GmbH*** für die Bereitstellung von Evals ihrer PKCS#11-Bibliothek, insbesondere bei Support und Entwicklung für umfangreiches Debugging sowie bei Herrn Georg Schmidt für Informationen aus Entwicklersicht und Einblicke in die Herangehensweisen bei XML- und PDF-Signaturen,

Herrn ***Dipl.-Ing. Nelson Langkamp*** (Langkamp-IT e.U.) für die freundlichen Auskünfte zur Andoid-Anwendung sPDF sowie bei

Herrn ***Daniel Konrad*** (A-SIT) für die zahlreichen technischen Auskünfte.

Wien, im September 2011

Gernot Schmied & Wolfgang Fabics

1. Gegenstand und Inhalt

1.1. Ausgangssituation

Die folgenden Wahrnehmungen und Kritikpunkte bilden die Basis unserer Untersuchungen und werden in weiterer Folge behandelt:

- Der ggw. *status quo* frustriert Laien, Power-User und Informatiker gleichermaßen.
- Die Marktkommunikation war und ist unglücklich (willkürliche Vermischung von Funktionen, Zertifikatsträgern, rechtlichen und technischen Begrifflichkeiten, Kunstwörtern).
- Benutzerfreundlichkeit (*Usability*) und *Complexity Hiding* sind nicht annähernd befriedigend gegeben, die Produkte bewegen sich nach wie vor zu wenig in der Begriffswelt und „*Comfort Zone*" der Anwender.
- Der Fehlerfall überfordert Laien völlig (komplexe Abhängigkeiten und Anforderungen), die Informationssuche ist unzumutbar und führt rasch zu Entwickler-Foren.
- notorische branchenspezifische Probleme (z.B. Online-Banking);
- zunehmende, aber noch zu geringe Verbreitung der mobilen Signatur als gute Alternative zur lokalen BKU;
- unreflektierte Microsoft-Windows-Lastigkeit an der Marktrealität vorbei, beharrliches Verweigern der Marktsignifikanz von Linux und Apple OS X bzw. „Firefox"- und „Chrome"-Browsern
- Der isolierte Ausgangspunkt „hoheitliche Spezialanwendungen" war nicht gerade visionär gewählt, dies äußert sich auch symptomatisch in der historisch bedingten lange dominierenden Diktion „Bürgerkarte" bzw. „Bürgerkartenfunktion" und auch „Verwaltungs"- bzw. „Amts-Signatur". Der Begriff „e-Card" hingegen wird eher nur mit Funktionen der Sozialversicherung in Zusammenhang gebracht.
- Ebenfalls außer Acht gelassen wurde die Frage nach ausreichenden Anreizen für Produktentwickler und das rechtzeitige Bereitstellen einer vollwertigen Programierschnittstelle für Web-Anwendungen und aus aktueller Sicht für die Verwendung der mobilen Handy-Signatur in Google Android und Apple IOS Apps.

Fazit: Akzeptanz und damit das Erreichen einer kritischen selbsttragenden Menge entstehen nicht durch selten benötigte Spezialanwendungen (z.B. e-Voting alle paar Jahre) sondern durch ständigen unmittelbaren Nutzen im täglichen privaten und unternehmerischen Alltag. Diese kritische Menge im engeren Sinne ist nach wie vor nicht erreicht (und nicht mit dem bloßen Besitz einer e-Card zu verwechseln).

1.2. Herangehensweise

Eine historische Aufarbeitung der technischen, rechtlichen und politischen Design-entscheidungen, Rahmenbedingungen, Weichenstellungen, realen Sachzwänge und (Fehl-)Entscheidungen in der Geschichte der elektronischen Signatur in Österreich

wird mit einigen wenigen Ausnahmen absichtlich nicht angestellt. Ebensowenig wird der durchaus berechtigten Frage nachgegangen, ob aus Fehlern oder der Anwenderablehnung überhaupt Lehren gezogen wurden. Wir möchten auch nicht den Eindruck erwecken, eine Position á la „im Nachhinein ist man immer gescheiter" einnehmen zu wollen.

Vielmehr betrachten wir ausschließlich die Gegenwart und blicken in die antizipierbare Zukunft einer Informationsgesellschaft im fundamentalen Umbruch:

- ***Änderung des Kommunikationsverhaltens*** weg von E-Mail und SMS hin zu sozialen Netzen, weg von proprietären E-Mail-Clients hin zu Web 2.0/HTML 5 bzw. Cloud-Applikationen wie Google Docs oder Microsoft Office 365[1];
- ***Stagnation bei der Weiterentwicklung von E-Mail***, z.B. ist nach wie vor keine Durchsetzbarkeit von Sende- und Empfangsbestätigungen gegeben, eine Motivation für durchgängige server-seitige Implementierung standardisierter E-Mail-Mehrwertdienste ist nicht erkennbar[2].
- ***Online/Cloud Storage und Netzwerk-Freigaben (Shares)*** wie dropbox, icloud, box.net oder Wuala ersetzen zunehmend den bislang üblichen Dateiversand via E-Mail-Attachments;
- ***Mobilität, Wireless Access & „Always-On Internet":*** Smartphones, Tablet-PCs und die neueste Generation von Notebooks verdrängten überraschend rasch den klassischen Desktop-PC; die meisten Unternehmen ersetzen ihre ausrangierten Desktops nicht mehr durch Standgeräte. Die neuen mobilen Geräte verfügen i.d.R. nicht über die Möglichkeit, einen Chipkartenleser anzuschließen bzw. anzusprechen und tlw. auch nicht über die des SMS-Empfangs (z.B. Apple iPad2), bedeuten aber auch neue Sicherheitsimplikationen, die es zu bedenken gilt.
- ***Device-SIMs ersetzen Scheckkarten-Slots*** (Miniaturisierung), Geräte mit mehreren SIM-Slots betreten den Markt; diese Entwicklung gilt es zu beobachten.
- ***32bit-Betriebssysteme sind de facto Geschichte***, das Interworking zwischen 64bit-Betriebssystemen und 32bit-Applikationen ist nicht immer problemlos gegeben.

Diese Studie untersucht auch aus diesen Gründen ausschließlich Lösungen, die

- in aktuellen 64bit-Betriebssystem- und Software-Versionen funktionieren,
- aktuelle Versionen der in Österreich verbreiteten Signaturkarten (A-Trust & e-Card) unterstützen,
- interessierten Laien zumutbar sind und
- auf absehbare Zeit unterstützt und weiterentwickelt werden.

Wenngleich zur Beurteilung der untersuchten Hard- und Software-Komponenten tlw. höchst umfangreiche technische Analyse- und Debugging-Aufwände getätigt werden mussten, so sind diese naturgemäß ebenso produktabhängig wie zeitlich volatil und bergen i.d.R. keine weiterführenden Erkenntnisse für die adressierte Leserschaft.

1 Dies hat Auswirkungen auf die Möglichkeit des Einsatzes von S/MIME mit Signaturkarten und der mobilen (SMS-gestützten) Signatur. Weder S/MIME noch PGP haben sich als Standard-Feature in den web-basierten E-Mail-Frontends durchgesetzt.

2 beeindruckend vollständige Ausnahme: der e-Post-Dienst der Deutschen Telekom

Deshalb ist die Detailbehandlung bestehender technischer Probleme ausdrücklich nicht Gegenstand des vorliegenden Dokuments.

Eine vollumfängliche Abdeckung beliebiger Konstellationen von Benutzeroberflächen, Anwendungen und Betriebssysteme wäre eine Lebensaufgabe. Wir haben uns daher auf wesentliche Einsatzszenarien und Werkzeuge beschränkt, dies ohne Anspruch auf Vollständigkeit. Die Erkenntnisse und Empfehlungen, zu denen die Studie gelangt, beruhen notwendiger Weise auf den tatsächlichen Gegebenheiten und Sachzwängen der ggw. österr. Signaturansätze – die völlige Abkehr von vorhandenen Architekturen oder die Implementierung visionärer, aber andersartiger Konzepte halten wir zum gegebenen Zeitpunkt für unrealistisch.

2. Rahmenbedingungen

2.1. Use Cases

2.1.1. Einleitung

Die verbreitete Verwendung elektronischer Signaturen kann unserer Ansicht nach nur auf dem offensichtlichen Nutzen und der einfachen Handhabung für den durchschnittlichen Benutzer beruhen – sobald dies erreicht ist, würden sich voraussichtlich die eingesetzten Verfahren und Komponenten von alleine weiterentwickeln, da hierfür ein Anreiz aus dem Markt gegeben wäre.

Im Gegensatz dazu wurde in Europa bislang der genau gegenteilige Ansatz verfolgt, nämlich:

- Zuerst Schaffung einer rechtlich-technischen Rahmenstruktur, die ohne Rücksicht auf vorhandene Lösungen und Bedürfnisse den Stand der Wissenschaft (nicht den Stand der allgemein verfügbaren Technik) als „*de facto*"-Standard vorschreibt;
- dann die Konzeption von Lösungsszenarien, die alleine den Nutzen für die öffentliche Verwaltung in den Vordergrund stellen;
- danach isolierte nationale technische Entwicklung der dazu erforderlichen Methoden und Betriebsmittel auf überaus eingeschränkter Anbieterbasis und erst zuletzt
- die Erkenntnis, dass mit diesen Lösungen ein Anreiz weder für den Bürger zur täglichen und selbstverständlichen Nutzung noch für die Industrie zur laufenden (Weiter-)Entwicklung von Produkten geschaffen wurde.

Der Schwerpunkt unserer Betrachtung liegt daher auf dem generischen Nutzen und den Anreizen für Unternehmer und Privatpersonen, elektronische Signaturen in der täglichen Kommunikation einzusetzen – nicht auf dem Dialog mit der Hoheitsverwaltung und Spezialanwendungen für besondere Berufsgruppen (Rechtsanwälte, Notare, Ziviltechniker, Gerichtssachverständige, Ärzte u.a).

2.1.2. essentielle Use Cases

Die Berücksichtigung der folgenden Anwendungsszenarien erachten wir als maßgeblich, wenn ein Signaturkonzept dem Anspruch an allgemeine und zeitgemäße Nützlichkeit genügen soll:

- ***rund um E-Mail:*** Unterschrift & Verschlüsselung von E-Mails & PDF-Attachments, Inanspruchnahme von E-Mail-Zustelldiensten
- ***rund um PDF:*** Unterschreiben und Verschlüsseln von PDF-Dokumenten & -Formularen, Verifikation & Entschlüsselung, Zeitstempelung & Ortsangabe
- ***Verwendung direkt aus Office-Umgebungen:*** Microsoft Office/365, LibreOffice, Google Docs
- ***Verwendung in Browsern:*** MS Internet Explorer, Firefox, Google Chrome u.a.

- ***rund ums Geld:*** Internet-Banking & Kreditkarten, e-Payment[3], e-Rechnung, FinanzOnline
- ***e-Verträge*** (Mehrfachzeichnung) & Vollmachten
- ***Zugang zu Web-Portalen*** und signatur-aktivierten Internetseiten
- ***Crypto-Storage:*** TrueCrypt hat sich als *„de facto*"-Standard etabliert.
- ***Online-Storage*** (spezifische österr. Angebote): e-Tresor, Manz Storage u.a.
- ***Cloud-Storage:*** box.net, wuala, dropbox, iCloud
- ***mobile Signatur*** auf Smartphones & Tablets (Apple IOS, Google Android u.a.)

Es ist hier strikt zwischen dem eigentlichen *Use Case* und den dafür erforderlichen *„Mitteln zum Zweck"* zu unterscheiden. Unter letztere fallen beispielsweise:

- Signaturprinter als Abstraktionslayer für Office-Suiten
- PDF/A-Validator
- Certificate-Stores & Schlüsselbunde
- BKUs & SecurityLayer
- Secure Viewer

2.2. Usability & Complexity-Hiding

Bürgerkartenumgebungen (bzw. deren GUIs und Anwendungen) zeichnen sich immer noch durch wenig durchdachte Menüführungen („Karten aktualisieren", Zugriffseinstellungen, XML-Signatur) und Darstellungen aus, die dem Anwender entweder nichts sagen oder diesen verwirren (schlecht gewählte Begriffe, Anordnung, Überfrachtung, Symbole, Verifikationsbestätigungen).

Die Aufgabe, Komplexität vor dem Benutzer zu verbergen und sich in seine Begriffswelt, Aufmerksamkeitsspanne und Frustrationstoleranz zu versetzen, seine kognitive Wahrnehmung zu antizipieren und dementsprechende Workflows zu bauen und zu testen, wurde unbefriedigend gelöst. Dies äußert sich mitunter in kryptischen Fehlermeldungen, die darauf folgende mühsame Suche nach Workarounds in Foren trübt nachhaltig das Benutzungserlebnis.

Benutzer werden zu Entscheidungen aufgefordert (z.B. *Trust Levels*), die sie mangels fachlichem Verständnis nicht sinnvoll treffen können. Hier sind durchdachte Default-Vorgaben gefragt. Kein typischer Benutzer versteht den Unterschied zwischen RSA- und ECDSA-Zertifikaten bzw. die sich daraus ergebenden Implikationen von fortgeschrittener und qualifizierter Unterschrift, ebensowenig die Struktur oder Attribute von X.509-Zertifikaten.

3 Man denke an aktuelle Entwicklungen wie Google Wallet via NFC (*Near Field Communications*).

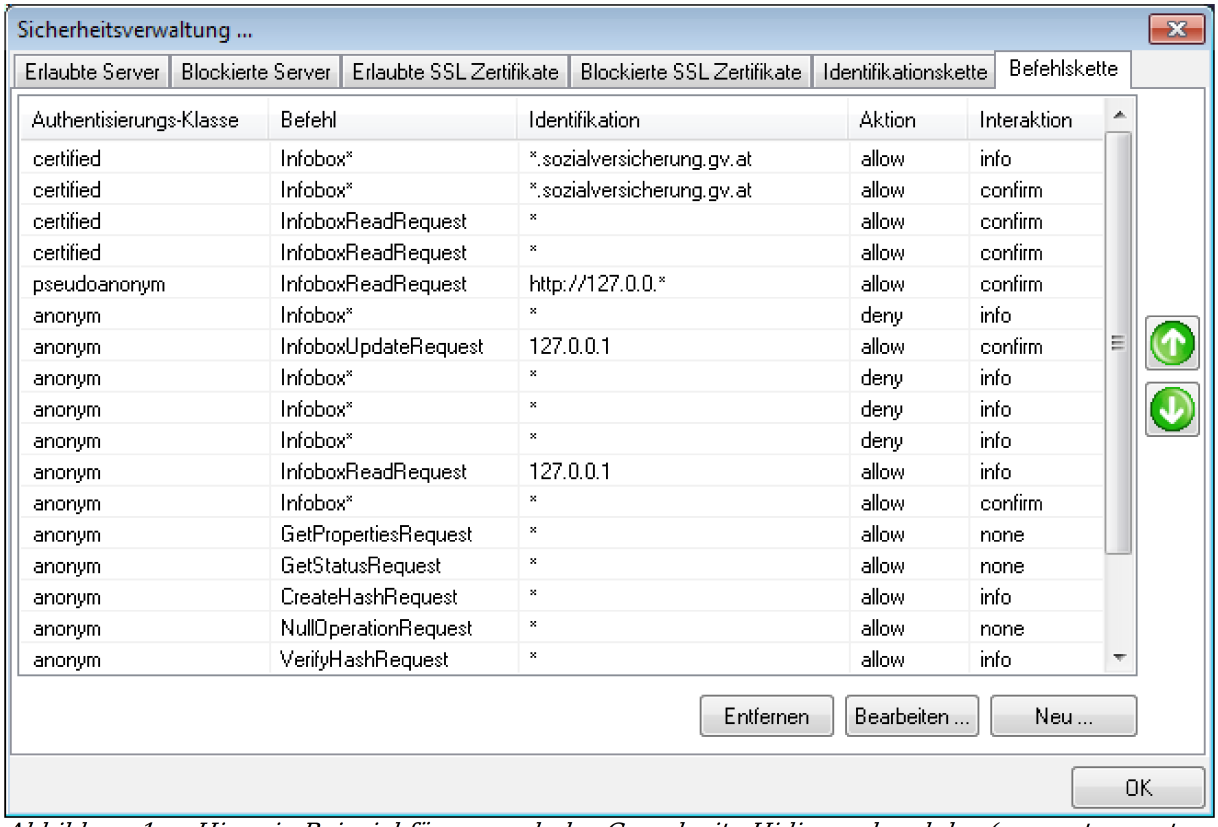

Abbildung 1: Hier ein Beispiel für mangelndes Complexity Hiding anhand des (ansonsten gut gelungenen) TrustDesk Pro von IT Solution. Selbst dem versierten Benutzer erschließen sich Zweck und Informationsgehalt des Dargebotenen nicht auf den ersten (oder auch zweiten) Blick - cui bono?

Unbefriedigend ist auch die Vielzahl an manuellen Schritten zum Export/Import von Karten-Zertifikaten bzw. Stamm-/Zwischenzertifikaten sowie die leidige Konfiguration vieler Software-Produkte und deren Interaktion mit Plugins (als umfassendes Beispiel sei hier die Adobe Suite genannt).

Installationspakete für die BKUs von A-Trust und IT Solution haben die Situation mittlerweile zwar etwas verbessert, die angesprochenen grundlegenden Defizite bestehen jedoch nach wie vor.

2.3. Token (Chipkarten)

Die aktuell meistverbreiteten österr. Signaturkarten-Generationen genießen unterschiedlich gute Unterstützung in Betriebssystemen und Standard-Anwendungen. Abgesehen von unterschiedlich implementierten Karten-Schnittstellen besteht der hautsächliche Unterschied in den enthaltenen Zertifikaten bzw. Schlüsselpaaren für die qualifizierte Signatur:

- Die A-Trust-Karten (a.sign premium) enthielten von Anfang an neben dem ECDSA-Schlüsselpaar für die qualifizierte Signatur auch ein RSA-Schlüsselpaar, das für die Verwendung durch Standard-Applikationen vorgesehen ist.

- Die e-Card hingegen enthielt bis zur G2 zwei ECDSA-Schlüsselpaare - mit dem zweiten Schlüsselpaar hätte man zwar div. Standard-Anwendungen bedienen *dürfen*[4], mangels Applikationsunterstützung der 192bit-ECDSA-Schlüssel waren diese jedoch *de facto* nicht nutzbar. Erst mit der e-Card G3 steht ebenfalls ein RSA-Schlüsselpaar zur Verfügung.

Gemeinsam ist all diesen Karten, dass erst in der jüngsten Generation der Wechsel auf ECDSA-Schlüssel auf Basis von 256bit-Kurven vollzogen wurde; die Tatsache, dass dieser Algorithmus von der NSA (im *„Suite B" Cryptography Standard*) offiziell empfohlen und für die Verwendung bis zur Sicherheitsstufe „SECRET" zugelassen wird, lässt die zunehmende weltweite Implementierung in Betriebssystemen und Standard-Applikationen erwarten:

- ACOS 03 (A-Trust Karten) ECDSA 192bit
- ACOS 04 (A-Trust Karten) ECDSA 256bit
- STARCOS G2 (e-Cards ohne SV-Zeichen in Brailleschrift) ECDSA 192bit
- STARCOS G3 (e-Cards mit SV-Zeichen in Brailleschrift) ECDSA 256bit

Es steht dringend zu hoffen, dass nunmehr keine Karten mehr neu ausgegeben werden, deren ECDSA-Zertifikate auf den so gut wie nicht unterstützten 192bit-Kurven beruhen. Auch halten wir es für unabdingbar bei künftigen Deployments, dass nur Chipkarten berücksichtigt werden, für die unter den drei betrachteten Betriebssystemen

- eine getestete (!) Unterstützung durch die gängigen Kartenleser(-treiber) gegeben ist und
- entsprechende PKCS#11-Module angeboten werden[5].

Wenngleich Projekte wie OpenSC oder die Entwicklungen von A-Trust und IT Solution gezeigt haben, dass solche Module (mit erheblichem Aufwand und entsprechenden Entwicklungszyklen) auch von Dritten in guter Qualität bereitgestellt werden können, so sollten diese aller Möglichkeit nach vom Kartenhersteller selbst kommen.

2.4. Signaturformate

2.4.1. CMS-Signatur (*PKCS#7 Detached Signature*)

Dieses Signaturverfahren ist das universell verwendbare für jedes beliebige zu signierende Dateiformat, da die Signatur in einer gesonderten Datei in einem

4 Dies liegt am Konzept des *„Dual Keying"*, nach dem für Verschlüsselungszwecke ein anderer Schlüssel als für die qualifizierte Signatur verwendet werden muss. Zum einen darf das qualifizierte Schlüsselpaar nicht exportiert werden (→ kein Backup möglich, daher für Verschlüsselung suboptimal), zum andern gelten aus Sicherheitsgründen unterschiedliche Regeln für Länge und Abfrage des PINs.

5 In Entwicklerkreisen wird bisweilen die Ansicht vertreten, dass PKCS#11 insbesondere für qualifizierte Signaturen keine zeitgemäße Schnittstelle mehr sei; dies mag in rein technischer Hinsicht sogar zutreffen. Jedoch ist PKCS#11 nach wie vor *die weltweit implementierte* Karten-Schnittstelle für Applikationen schlechthin - ohne ihre Bereitstellung ist der *Use Case* „Nutzung der elektronischen Signatur in Standard-Anwendungen" *de facto* gestorben.

genormten Format (entsprechend CMS, der *Cryptographic Message Syntax* gem. PKCS#7) abgelegt wird[6].

Der Hashwert wird dabei einfach über die gesamte Datei gebildet, sodass jede kleinste Veränderung an dieser Datei zur Ungültigkeit der Signatur führt, was theoretisch von Vorteil für die Sicherheit ist, praktisch aber in bestimmten Konstellationen bisweilen ein Nachteil sein kann (z.B. wenn in der Datei veränderliche Metadaten enthalten sind, die den eigentlichen Nachrichteninhalt nicht berühren, etwa Zeichencodierungen) - hier bietet XMLDsig ggf. Vorteile.

Einige BKUs bieten (bzw. boten) die Funktion „CMS-Signatur" an - allerdings nur für die von ihnen unterstützten Dateiformate (i.d.R. XML, HTML und ggf. PDF); diese Einschränkung hat gewissermaßen historische Gründe und ist in der (mittlerweile rechtlich obsoleten) Verwendung des eingebauten Secure Viewers begründet.

Anzumerken bleibt, dass vmtl. nicht viele Benutzer mit einem Menüpunkt wie „CMS-Signatur" etwas anfangen können. Eine Bezeichnung wie „Unterschrift in getrennter Datei erstellen", ggf. sogar mit einem aussagekräftigen Hilfetext, wäre hier deutlich sinnvoller.

2.4.2. XML-Signatur (XMLDsig)

Das Container-Format von XMLDsig ist in XML codiert und vom „*World Wide Web Consortium*" (W3C) standardisiert. Was dieses Format vielseitig nützlich macht, ist der Umstand, dass es in drei Ausprägungen angewendet werden kann:

- ***enveloped:*** Hierbei ist der Signaturblock in die Struktur einer (XML-)Datei eingebunden, die Signatur ist also Teil der signierten Datei.
- ***enveloping:*** Umgekehrt kann auch die signierte Datei in den Signaturcontainer eingebettet werden; hier sind also die signierten Daten Teil der Signaturdatei selbst.
- ***detached:*** Dieses Verfahren ist - wie die CMS-Signatur - für beliebige Dateiformate verwendbar, da die signierten Daten im Container nur referenziert (verlinkt) werden.

Tatsächlich ist XMLDsig die in österr. Signaturlösungen wohl am häufigsten eingesetzte Signaturform (wenngleich sie dem Benutzer meist verborgen bleibt); ihre intrinsischen Möglichkeiten zur gleichzeitigen Signatur mehrerer Dateien und deren eindeutigen Referenzierung macht man sich z.B. auch bei Lösungen zum „sicheren Verketten" gem. BVergG zu nutze.

Anzumerken ist aber auch, dass in den heute verbreiteten Standard-Applikationen XMLDsig so gut wie keine Rolle spielt.

2.4.3. PDF-Signaturen

Zur Erklärung der unterschiedlichen Signaturformate für PDF-Dateien sei die Information von http://www.buergerkarte.at zitiert:

6 Die typische Dateinamenserweiterung einer solchen Signaturdatei ist *.p7s („*PKCS#7 Signature*"), die CMS-Signatur ist aber auch das von S/MIME verwendete Verfahren (hier findet man dann **Attachments** vom Typ *.p7m, „*PKCS#7 Mail*").

Alle Arten der Signatur beruhen auf dem allgemeinen Standard PKCS#7. Darauf aufbauend haben sich jedoch drei[7] *verschiedene Signaturen entwickelt:*

- ***Adobe-Signatur:*** *Die eigentliche Signatur steht in einem speziellen unsichtbaren Bereich der PDF-Datei (das so genannte Signature Dictionary im Incremental Update Block). Die Anzeige erfolgt im Unterschriften-Fenster in Adobe Reader.*
- ***PDF-AS:*** *Die Signatur steht im Signaturblock; dieser wird im Dokument ganz normal angezeigt. Darüber hinaus gibt es keinen unsichtbaren Signatur-Bereich.*

PDF-AS (von „Amtssignatur") bildet hier insofern ein reines *Austriacum*, als die Signatur *de facto* in das PDF-Dokument gedruckt wird (also Teil der Content-Daten ist). Das ist der Grund, warum es keinen unsichtbaren Signaturbereich gibt – und warum diese Art der Signatur auch nur von speziellen österr. Lösungen (i.d.R. online!) verifiziert werden kann[8]. Eine Unterstützung dieses Formats in weltweit genutzten Standard-Anwendungen ist daher u.E. nicht zu erwarten.

Näheres zu den Besonderheiten von PDF-AS findet sich in Punkt 3.2.1. zu „PDF-Over".

2.5. rechtlicher Rahmen

Eine detaillierte Betrachtung des rechtlichen Umfelds muss im Rahmen der vorliegenden Studie unterbleiben, dieses wurde z.B. in der netIdee-Studie 2007 „PGP-Zertifikate - des Signaturrechts vergessene Kinder?" am Beispiel der rechtlichen Verankerung von PGP als „fortgeschrittenes" Signaturverfahren eingehend behandelt[9]. Die ggstdl. eher technisch-betriebliche Betrachtung der Materie wäre jedoch nicht vollständig, ließe man die durchaus engen Wechselwirkungen zwischen der rechtlichen Verankerung, der technischen Entwicklung und der tatsächlichen Verwendung elektronischer Signaturen außer acht. Einige Kernaspekte sollen hier deshalb mitsamt ihren konkreten Auswirkungen aufgezeigt werden.

Die SigRL der EU (und in der Folge ihre nationale Umsetzung in verschiedensten Rechtsquellen) unternimmt den an sich durchaus lobenswerten Versuch einer Vereinheitlichung der unterschiedlichen nationalen Interpretationen der technischen und rechtlichen Wertigkeit elektronischer Signaturverfahren. Dabei bedient sie sich aber Methoden, die ihre tatsächliche Anwendung auf breiter Basis bereits von vorneherein nachhaltig behindern:

- ***Regelungsgegenstand:*** Es wird zwar beschrieben, welchen technischen und betrieblichen Anforderungen ein Signaturverfahren genügen muss, um in einem bestimmten Kontext als rechtssicher anerkannt werden zu können; jedoch fehlt eine klare Regelung, unter welchen Umständen eine elektronische Signatur jedenfalls anerkannt werden *muss*. Dies führt zur

7 Die dritte auf www.buergerkarte.at gelistete Signaturvariante, PKCS#7, wird in Punkt 2.4.1 „CMS-Signatur" behandelt, da sie sich für beliebige Dateiformate eignet und daher keine spezifische „PDF-Signatur" darstellt.

8 Weiters gibt es bei PDF-AS neben der üblichen „Binärsignatur" (der gesamten Datei) auch eine sog. „Text-Signatur", die zur Verifizierung eines signierten Text-Ausdruckes durch Abtippen entwickelt wurde - ein ausschließlicher *Use Case* der öffentlichen Verwaltung.

9 Auf Änderungen der Rechtslage seitdem wird jedoch im jeweiligen Kontext sehr wohl eingegangen.

optimalen Unsicherheit bei den Rechtsunterworfenen, da nicht a priori klar ist, welchen rechtlichen Nutzen ein bestimmtes (ggf. bereits eingesetztes) Signaturverfahren entfalten kann. Da solcherart der Verwendungsnutzen hochgradig unbekannt ist, darf es nicht verwundern, wenn Implementierungen bestenfalls zögerlich erfolgen oder aber in vielen Fällen einfach auf herkömmliche, jedoch – tatsächlich oder vermeintlich – rechtssicherere Verfahren aus der „Papierwelt" zurückgegriffen wird.

- ***Begriffswelt:*** Die von der „Signatur-Community" und im Signaturrecht verwendeten (und offenbar dafür neu erfundenen) Begriffe sind *per se* weder technisch noch alltagssprachlich aussagekräftig und waren zum Zeitpunkt ihrer Entstehung auch beileibe nicht in der IT-Fachwelt etabliert; ihre rechtliche Relevanz beruht alleine auf den dazugehörigen Legaldefinitionen. Dass diese Begriffe – zumindest in der deutschsprachigen Version der SigRL – auch noch höchst ungeschickt übersetzt werden (Warum kann man das engl. *signature* nicht allgemein verständlich als „Unterschrift" übersetzen? ... usw.), weist umso mehr auf ihre Entstehung in einem „Elfenbeinturm" hin und trägt zur Akzeptanz beim normalen Internet-Nutzer nicht wirklich bei. Abgesehen davon existiert in der IT-Welt ja bereits eine umfangreiche und v.a. in den Standard-Applikationen seit langem verwendete Nomenklatur, auf die hier keinerlei Rücksicht genommen wurde – Unklarheiten und Missverständnisse sowohl bei der GUI-Gestaltung als auch bei der Benutzung sind dadurch vorprogrammiert.
- ***Technologieneutralität:*** Wie üblich wurde versucht, die rechtlichen Festlegungen möglichst technologieneutral zu gestalten. So sinnvoll dieses Bemühen im Grunde ist, so fatal hat es sich auf die tatsächliche Diktion im Signaturrecht ausgewirkt: Konkret beschrieben wird ein Rechtsrahmen, der eindeutig und ausschließlich auf Signaturverfahren der X.509-Suite anzuwenden wäre. Um die dort gebräuchlichen und klaren Begrifflichkeiten zu vermeiden (*cui bono?*), bedient man sich aber Ausdrücken und Umschreibungen, die eine einfache Anwendung auch in diesem eingeschränkten Kontext unnötig erschweren und sogar innerhalb der „Signaturgemeinde" zu folgenschweren Missverständnissen und Fehlinterpretationen geführt haben: z.B. verwenden alle Signaturverfahren, auf die sich die in der SigRL beschriebenen Szenarien anwenden lassen, die Begriffe „öffentlicher Schlüssel" und „privater Schlüssel – wem dient die Einführung von „Signaturerstellungsdaten" und „Signaturprüfdaten" (zumal nicht einmal von vorneherein klar ist, ob z.B. PINs und Passwörter bzw. Passphrases vom Begriff der „Signaturerstellungsdaten" erfasst sind)?

3. Bürgerkartenumgebungen

3.1. Wesen & Darreichungsformen

Mit der „Bürgerkartenumgebung" wurde - analog zur österr. Rechtslage vor der SigG-Novelle 2009 - das Konzept einer integrierten Software-Lösung geschaffen, die ohne Abhängigkeiten vom Betriebssystem oder den Applikationen eine einheitliche Applikationsschnittstelle („Security Layer"), die relevanten Signatur- und Verwaltungsfunktionen sowie eine zuverlässige Anzeigefunktion der zu signierenden Dokumente („Secure Viewer") bereitstellt. Die wichtigsten der damit realisierten Anforderungen sind jedoch seit der SigG-Novelle tatsächlich nicht mehr gegeben:

- Notwendigkeit zur Zertifizierung der gesamten „Signaturumgebung" (es wird nur noch die „sicherer Signaturerstellungseinheit" (also die Chipkarte selbst) durch die Bestätigungsstelle zertifiziert,
- damit Wegfall der Zertifizierung (und zwingenden Verwendung) eines „Secure Viewers"; gem. EU-SigRL ist es auch nicht erforderlich, dass die eingesetzten Signaturkomponenten die Anzeige der zu signierenden Daten selbst *ermöglichen* oder gar *erzwingen*, sie dürfen diese nur *nicht verhindern* (z.B. durch die entsprechende Standard-Applikation)[10].

Was jedoch bleibt, ist die Notwendigkeit für österreichische eGov-Applikationen, die auf dem Zertifikatsträger aufgebrachte Personenbindung auslesen zu können (diese erst macht eine Chipkarte mit qualifiziertem Zertifikat zur "Bürgerkarte") - und diese Funktion ist dzt. (und auf absehbare Zeit auch weiterhin) nur in den typisch österreichischen BKUs realisiert. Diese Anforderung ist jedoch vorwiegend für die elektronische Authentifizierung im österr. Verwaltungsverfahren von Relevanz und steht eigentlich nicht im direkten Zusammenhang mit der allgemeinen Verwendung elektronischer Signaturen.

Bürgerkartenumgebungen können in den folgenden Ausprägungen eingesetzt werden:

- ***lokale BKU*** – viele lokale Fehlerquellen, komplexe Abhängigkeiten
- ***Online-BKU*** – läuft im Browser, i.W. nur noch der Kartenleser-Anteil lokal
- ***mobile Signatur*** (Handy-Signatur) – keine lokalen Komponenten mehr erforderlich, tauglich für Smartphones und Tablets

Gemeinsam ist allen drei Varianten, dass mit dem qualifizierten Zertifikat keine Verschlüsselungsfunktionen bereitgestellt werden; dies ist eine Sicherheitsmaßnahme der qualifizierten Zertifikate selbst („*Dual Keying*") und liegt nicht in der Funktion der BKU begründet. Stellt ein BKU-GUI dennoch Verschlüsselung bereit, so geschieht dies mit dem RSA-Zertifikat auf der Chipkarte (und steht deshalb z.B. mit der Handy-Signatur nicht zur Verfügung)[11].

10 ein gelungenes Beispiel dafür ist der eidgenössische *LocalSigner* Client; dieser erlaubt die Auswahl eines externen Betrachters (z.B. Adobe Reader) zusätzlich zur internen Vorschau

11 Bei der Handy-Signatur befindet sich die SSCD samt dem qualifizierten Zertifikat in der „kontrollierten Systemumgebung" des ZDA, die Schnittstelle Handy-Tastatur → SMS ersetzt *de facto* nur die PIN-Eingabe am (nicht vorhandenen) Kartenleser.

3.2. integrierte Bürgerkartenumgebungen

Generell können auf Windows-Plattformen nur die integrierten Bürgerkartenumgebungen von A-Trust und IT Solution empfohlen werden, ebenso die MOCCA BKU (Java Webstart). MOCCA ist ggw. unter Linux und Apple OS X die einzig funktionierende BKU und somit alternativenlos. Beim MOCCA-GUI ist zumindest der Versuch erkennbar, Komplexität vor dem Benutzer zu verbergen.

3.2.1. PDF-Over

PDF-Over stellt ggw. die einzige funktionierende und auch weiterhin gewartete PDF-AS Standalone-Anwendung dar, leidet jedoch unter archaisch-lieblosem Design sowie mangelndem Komfort der Anwendungen.

Kritikpunkte an PDF-Over:

- Warum soll der Benutzer mit dem Unterschied zwischen Text- und Binärsignatur überhaupt belastet werden (XML-Signaturen sagen ihm ohnehin nichts)? Für PDF sollte grundsätzlich nur die Binärsignatur verwendet werden.
- Warum ist das Signaturfeld nicht einfach als Box veränderbar („Rahmen ziehen")?
- Warum gibt es keinen Dialog für Bezeichnung und Ort der Zieldatei („speichern unter...")?
- Warum gibt es keine „Einstellungen" im Programm, wozu ein getrenntes Konfigurationsprogramm?
- keine benutzerfreundlichen Installationspakete für Linux (.deb, .rpm) vorhanden; hier böte sich ein Repository für automatische Updates an.
- keine Prüfmöglichkeit in PDF-Over selbst (Signaturverifikation nur online möglich, jedoch nicht einmal ein Link ist hinterlegt)

Fazit:

- Die grundsätzliche Problematik des Uploads von Dateien via Browser zur PDF-AS Verifikation wurde nicht berücksichtigt, dies ist für vertrauliche Daten völlig inakzeptabel.
- Die generelle Abkehr von ECDSA- bzw. PDF-AS Adobe-Plugins wäre als Architekturoption für uns jedoch nachvollziehbar – ständig wechselnde Schnittstellen, jährliche Major-Release-Wechsel und bislang seitens Adobe eher Desinteresse an ECDSA. Umso mehr empfiehlt sich die Unterstützung von PKCS#11 als Standard-Schnittstelle auch für österr. Signaturkarten.
- Nicht gerade glücklich sind auch die bildliche Aufbereitung der Signaturbox und der Umfang der angezeigten bzw. überhaupt vorhandenen Attribute. Es kann vor Unterschrift kein Signaturzweck angegeben werden; dies macht eine Unterscheidung zwischen Transportversiegelung und elektronischer Willenserklärung unmöglich, Vertretungsvollmachten wären ggf. ebenfalls einzublenden.
- Aufgabe der sinnlosen textuellen PDF-Signatur-Variante zum Zwecke *„der Rekonstruktion einer Signatur von einem Papierausdruck*" – dies führt das Konzept der digitalen Datei mit digitaler Unterschrift sowohl legistisch wie

auch technisch *ad absurdum*.

3.2.2. A-Trust a.sign Client & BKU

- nur für Windows verfügbar (*Full Installer* oder Einzelkomponenten)
- Die Linux-Version des a.sign Clients ist in unbrauchbarem Beta-Stadium, die PKCS#11 Library wird vom System und den Applikationen nicht erkannt.
- grundsätzlich zufriedenstellende Lösung für Windows
- Sehr positiv fällt die Beigabe der PKCS#11-Library für Windows auf.
- Das a.sign ECDSA-Plugin für Adobe führt unter Windows 7 (64bit) mit Adobe Acrobat 9.4 reproduzierbar beim Unterschreiben zum Absturz, die Verifikation funktioniert jedoch.

3.2.3. IT Solution TrustDesk Pro

- nur für Windows verfügbar (*Full Installer*)
- grundsätzlich zufriedenstellende Lösung für Windows, weit verbreitet für Spezialanwendungen wie z.B. e-Vergabe oder Urkundsarchive
- beinhaltet gut funktionierendes ECDSA-Plugin für Adobe
- nur in aktueller „Pro“-Version 3.x empfohlen

3.2.4. BDC HotSign

- nur für Windows verfügbar
- nur in „Premium“-Variante überhaupt sinnvoll (e-Card-Unterstützung, Adobe-Plugin, PDF-Feature)

3.2.5. LocalSigner der Schweizer „Open eGov”-Initiative

Ein Paradebeispiel für eine gut durchdachten Anwendung zur PDF-Signatur ist der „*LocalSigner*” der „Open e-Gov”-Initiative der Schweizerischen Eidgenossenschaft. Wir möchten den Leser nachdrücklich einladen, dieses überaus gelungene Stück Software auszuprobieren und selbst den Vergleich mit den österr. Alternativen anzustellen[12].

In diesem Zusammenhang muss auch lobend die aufgeräumte und mit Liebe zum Detail gestaltete Web Page der „Open eGov”-Web-Plattform unserer Nachbarn erwähnt werden, die einen wohltuenden Kontrast zur unbefriedigenden und „zerfledderten” Situation in Österreich darstellt.

12 Es mutet schon fast ironisch an, dass eine Lösung, die nichts von österr. Implementierungen weiß, mittlerweile das von den Autoren bevorzugte Programm für den Umgang mit den RSA-Zertifikaten der österr. Chipkarten geworden ist; sie funktioniert besser und intuitiver als die meisten frei verfügbaren österr. Lösungen, kommt bestens mit der generischen Schnittstelle der Adobe-Produkte zurecht und ist von intuitiver Benutzerfreundlichkeit und Einfachheit geprägt.

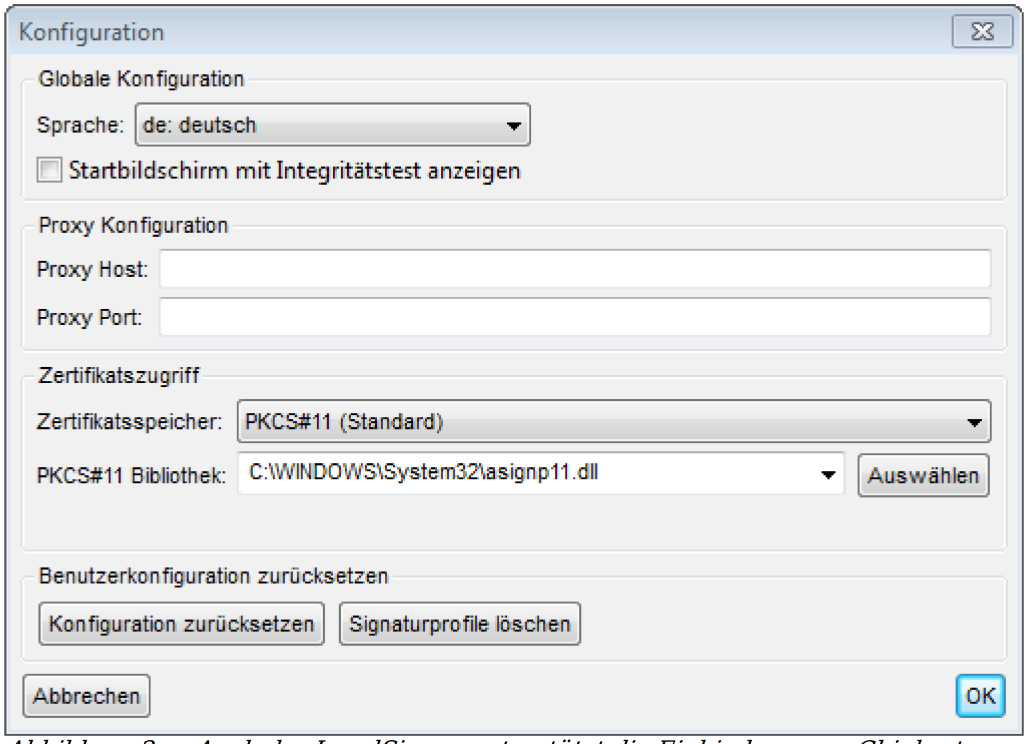

Abbildung 2: Auch der LocalSigner unterstützt die Einbindung von Chipkarten via PKCS#11, hier am Beispiel der A-Trust-Lib des a.sign-Client unter Windows 7.

Highlights:

- kommt bestens mit RSA-Zertifikaten zurecht
- benötigt keine Plugins für Adobe Reader/Acrobat
- ist auf allen Betriebssystemen verfügbar (benötigt jedoch dafür PKCS#11 Libraries) bzw. ist der Einsatz via CSP unter Windows alternativ möglich.
- kostenlose Zeitstempelung integriert
- Einzelsignatur und Mehrfachsignatur
- sowohl sichtbare als auch unsichtbare Signaturen möglich (verwendet die native Adobe-Signatur, vgl. dazu Punkt 2.4.3.)
- Möglichkeit des Anhängens von beliebigen PDF Dokumenten an die zu signierende Datei vor der Signatur
- exzellente integrierte PDF-Funktionen
- intuitiver Workflow
- bestens dokumentiert
- für die Zukunft integrierte Interaktion mit Zustelldienst geplant

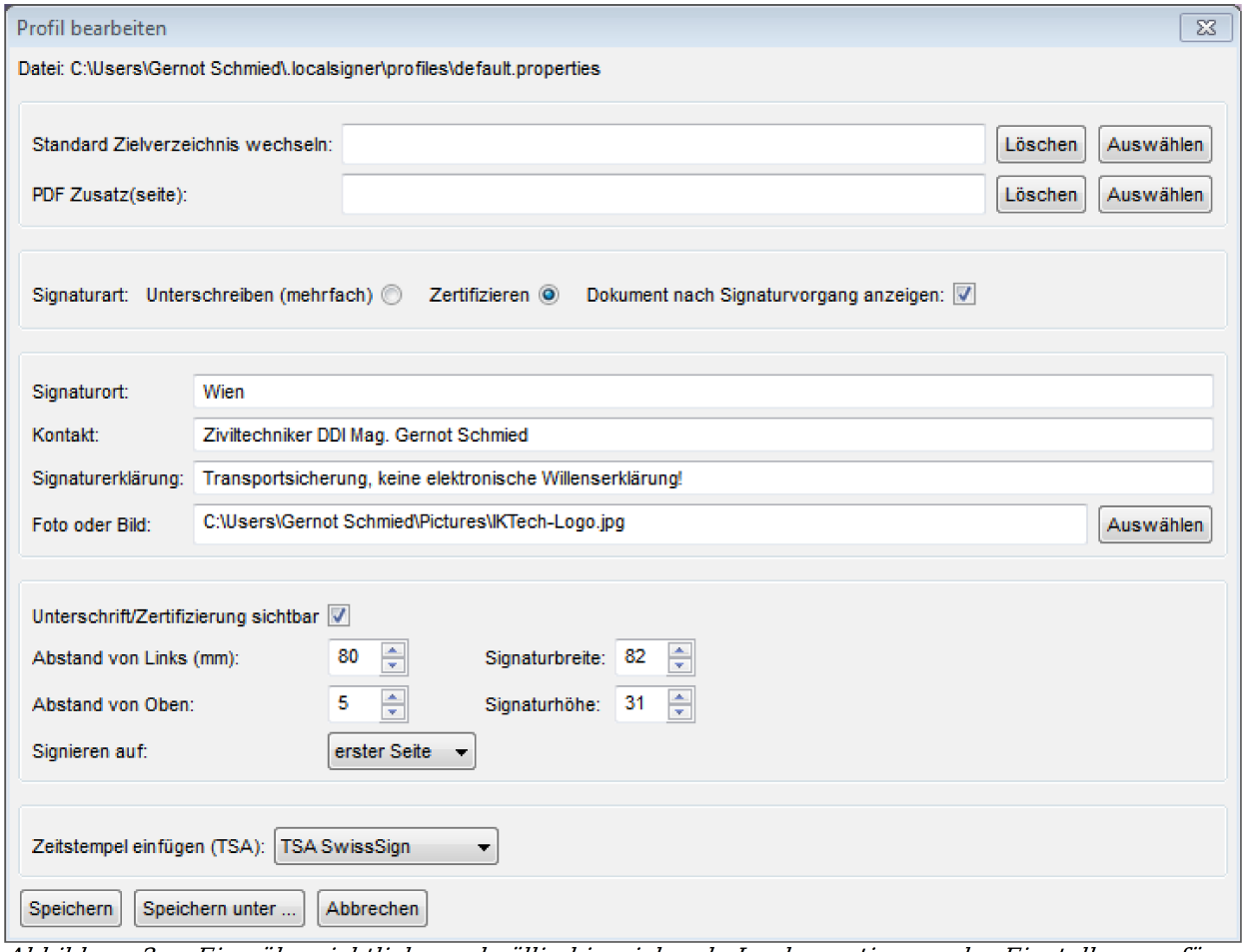

Abbildung 3: *Eine übersichtliche und völlig hinreichende Implementierung der Einstellungen für eine Adobe-PDF-Signatur mit LocalSigner (hier unter Windows 7), sehr sinnvoll die Verknüpfung mit einem Zeitstempeldienst. Wichtig ist aus unserer Sicht auch die "Signaturerklärung", die an jede Adobe-Signatur angefügt werden kann – dieses Konzept fehlt in den PDF-AS-Lösungen.*

4. Betriebssystem-Integration

4.1. Kartenleser, Treiber & PC/SC-Interfaces

Externe Kartenleser gibt es in verschiedenen Sicherheitsklassen (1 bis 3), mit und ohne eigenem Ziffernblock oder Display. In Notebooks integrierte Kartenleser weisen zwangsläufig keine dieser Zusatz-Features auf. Die Frage nach der Forderung einer eigenständigen Tastatur stellt sich insofern, da aus Implementierungssachzwängen das PC/SC-Interface betreffend oftmals der PIN dennoch via Computer-Tastatur eingegeben werden muss bzw. dies auch in Software umgestellt werden kann. Klasse 2 und 3 Leser sind zwar aus Sicherheitsgründen erstrebenswert (Risikoszenario Keyboard-Logger), sind jedoch nach ggw. Rechtslage kein zwingendes Erfordernis mehr für die Erstellung qualifizierter Signaturen.

PC/SC hat sich als Kartenschnittstelle durchgesetzt und bietet unter ***Windows***, Linux (pcscd) und Apple OS X (pcscd) eine stabile Implementierung, entweder integriert in BKU-Installationen oder über Betriebssystem-Pakete.

Unter ***Linux*** wird die PC/SC-Schnittstelle oberhalb der eigentlichen Kartenleser-Treibers durch mehrere Komponenten bereitgestellt:

- ***pcscd:*** Der PC/SC-Daemon verwaltet die einzelnen Treiber-Instanzen und die Verbindungen zwischen Applikationen und den Treibern;
- ***libpcsclite:*** Diese Bibliothek stellt das eigentliche PC/SC-Interface zwischen dem pscsd (bzw. den von ihm verwalteten Treibern) und den Applikationen bereit;
- ***pcsc-tools:*** Sammlung praktischer CLI-Werzeuge zum Debugging (Abfrage und Analyse der angeschlossenen Kartenleser und Karten, manuelles Absetzen von Leser- und Kartenbefehlen usw.)

Apple OS X verwendet für den Kartenleser-Zugriff ebenfalls den pcscd, das generische OS X „*SmartCard Services*"-Framework beruht auf CDSA/Tokend (*Token Daemon*). Ab OS X 10.7 „Lion" wurde CDSA von Apple aufgegeben, daher wird OS X Lion auch ohne Tokend ausgeliefert. Das Projekt wird nun vollumfänglich von http://smartcardservices.macosforge.org/ weitergeführt und kann auch für OS X Lion nachinstalliert werden. Ebenfalls zur Verfügung steht ein OS X Installationspaket des Projekts http://www.opensc-project.org/. Ggw. gibt es in beiden Projekten keine uns bekannte PKCS#11-Unterstützung der in Österreich verwendeten Signaturkarten[13].

4.2. Zertifikats-Management

Ein wesentlicher Usability-Aspekt ist aus unserer Sicht das Zertifikats-Management. Dieses unterscheidet sich unter den drei betrachteten Betriebssystemen maßgeblich:

- ***Microsoft Windows:*** Cryptographic Service Provider (CSP)
- ***Linux:*** „Verschlüsselung & Schlüsselbunde" (gnome-keyring + div. GUIs),

13 Die e-card G3 funktioniert mit MOCCA unter OS X 2.6 „Snow Leopard" nicht (wird nicht erkannt, vmtl. pcscd-Problem), es gibt allerdings Berichte, dass dies unter OS X 2.7 „Lion" behoben wurde.

daneben auch Certificate Stores der einzelnen Anwendungen (NSS und andere)

- ***Apple OS X:*** Schlüsselbundverwaltung (siehe dazu auch 4.1.1.)

Die Grundfunktionen bestehen im Import und Management von Zertifikaten (z.B. Trust-Level) und deren generischer Bereitstellung für Anwendungen. Das Verhalten einiger verbreiteter Standard-Anwendungen wurde dazu beispielhaft untersucht.

Ohne eine zentrale Zertifikatsverwaltung muss diese in den einzelnen Applikationen erfolgen - die Benutzerfreundlichkeit leidet dadurch erheblich. Alleine die Installation der erforderlichen *Root-* und *Intermediate-*Zertifikate ist dem Benutzer, sofern er diese synchron in mehreren Applikationen nutzen möchte, eigentlich nicht zuzumuten – zumal die GUIs in der Regel beim Import keine Möglichkeit der Mehrfachauswahl von Zertifikaten bieten.

Jedes Zertifikat muss daher einzeln importiert und der *Trust Level* eingestellt werden, alleine für die gängigsten österreichischen A-Trust-Zertifikate sind das also ca. 40 (!) einzelne Importvorgänge, und das gesondert für jede Applikation (vgl. auch Punkt 5.1.2.). Ein zentraler *Certificate Store* tut unserer Ansicht nach unbedingt not.

Auf der Suche nach Möglichkeiten, die nötigen Zertifikate unter Linux zentral zu verwalten (anstatt gesondert für jede Applikation) stößt man unweigerlich auf das GNOME-Keyring-Paket, das viele Distributionen von vorneherein mitbringen. Bereits der Name kann dazu verführen, die Möglichkeiten dieses Werkzeugs zu unterschätzen, denn:

- Obwohl eigentlich Teil der GNOME Desktop-Umgebung wird GNOME-Keyring durchaus auch z.B. mit KDE[14] erfolgreich verwendet.
- Anders als der Begriff „Keyring“ vielleicht vermuten lässt, verwaltet GNOME-Keyring nicht nur „Keyrings“ in der Diktion von PGP, sondern auch X.509-Zertifikate und Benutzer-Passwörter.

Die zentrale Komponente ist der gnome-keyring-daemon, der seine Management-Funktionen über drei verschiedene Komponenten bereitstellt:

- ***„login“*** verwaltet Passwörter und Authentizierungs-Informationen und stellt diese über entsprechende libraries anderen Systemkomponenten wie z.B. PAM zur Verfügung,
- ***„pkcs11“*** bietet eine Schnittstelle für PKCS#11-fähige Applikationen, die über die libgnome-keyring als „*Security Module*“ eingebunden werden kann. Solcherart erlangt die betreffende Applikation auf einfache Weise Zugriff auf alle vom gnome-keyring-daemon verwalteten Daten.
- ***„ssh“*** stellt einen vollwertigen Ersatz für einen anderweitigen SSH-Client (z.B. openssh) dar.

14 Linux mit KDE verwendet *Kleopatra* als integrierten OpenPGP- und X.509-Zertifikatsmanager. Kleopatra steht ebenfalls im Rahmen des Projektes http://www.gpg4win.org/ für Windows zur Verfügung und ist besonders attraktiv, wenn man beabsichtigt, mit X.509- und PGP-Zertifikaten zu arbeiten. Darüber hinaus wurden mit KDE keine Tests durchgeführt.

Es existieren verschiedene CLI-Tools[15], um Zertifikate zentral zu importieren und dem gnome-keyring-daemon zugänglich zu machen, sodass unter Verwendung der libgnome-keyring viele Standard-Applikationen auf einfache Weise mit einem und demselben Zert-Store arbeiten könnten.

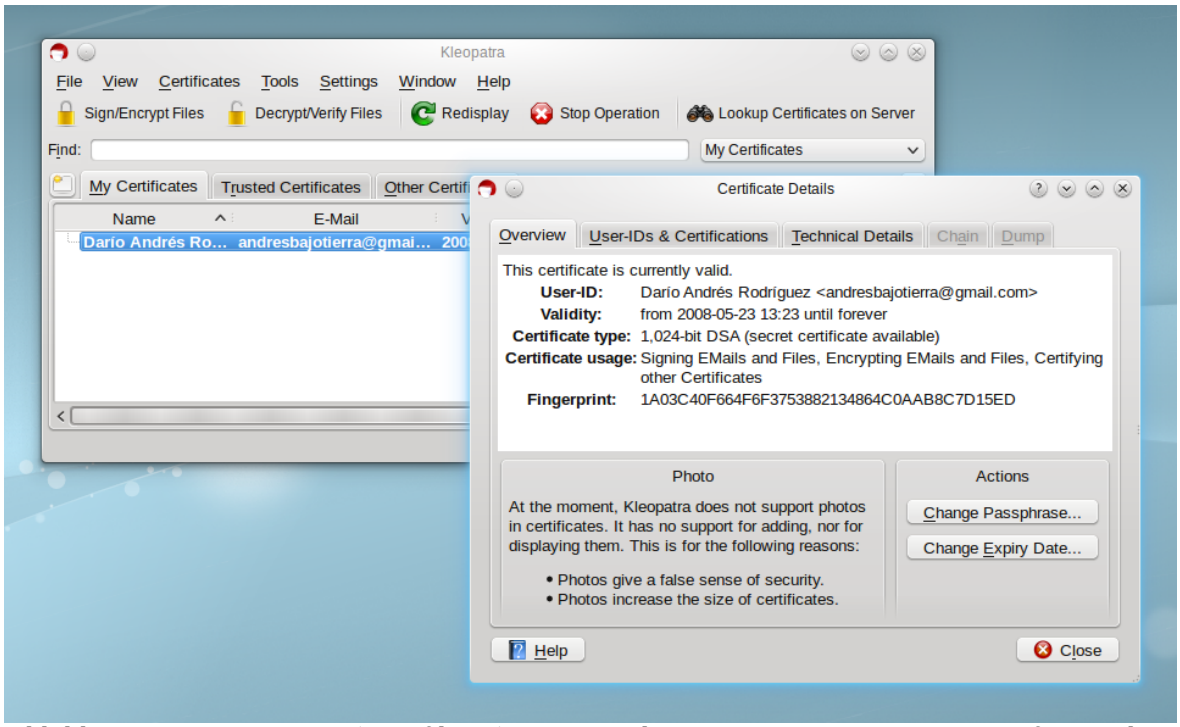

Abbildung 4: Der Keyring-(Zertifikats-)Manager Kleopatra unter Linux/KDE. Interface und Diktion sind zwischen PGP- und X.509-Zertifikaten wunderbar angeglichen. (Quelle: www.kde.org)

4.3. Desktop-Integration

4.3.1. Bürgerkartenumgebungen

Die Desktop-Integration der BKUs erfolgt in allen drei Betriebssystemen via Tray-Applikationen und erscheint grundsätzlich ausreichend, alle Einstellungen und Funktionen sind darüber erreichbar bzw. durchführbar. “Look & Feel” für den Benutzer bleiben jedoch diejenigen einer eigenständigen Applikation; eine Integration in die betriebssystemüblichen Dateimanager wird dringend empfohlen, wie bereits erfolgreich in der PGP-Welt vollzogen.

BKUs lassen sich problemlos manuell starten und stoppen bzw. bei Systemstart starten. Auch eine Installation unterschiedlicher BKUs macht keine Probleme, solange nicht der Versuch unternommen wird, sie gleichzeitig zu starten. Schwierigkeiten ergeben sich mitunter bei der Installation unterschiedlicher Plugins für Adobe Reader/Acrobat. Hier ist anzuraten, nur eine BKU einzusetzen und nicht zu mischen. Lediglich die Kombination MOCCA + PDF-

15 Ein komfortables GUI für die GNOME-Umgebung gibt es leider noch nicht (bzw. nicht mehr), da der „*GNOME Keyring Manager*“ seit GNOME 2.22 durch das ausschließlich für PGP konzipierte *Seahorse*-Projekt abgelöst wurde.

Over ist ggw. für alle Betriebssystem-Plattformen verfügbar (Java-Implementierung).

4.3.2. native Integration (Window Manager & Applikationen)

Eine vollständige native Integration von Signatur- bzw. Verschlüsselungsfunktionen in die Desktop-Umgebung (via Dateimanager) sucht man dzt. unter allen drei Betriebssystemen noch vergeblich. Insbesondere das funktionale Nebeneinander von PGP und X.509 ist stark applikationsorientiert und findet dzt. noch keine konzeptionelle Berücksichtigung, außer im integrierten Zertifikats-Manager *Kleopatra*. Probleme im kombinierten Einsatz von S/MIME und OpenPGP manifestieren sich oftmals auch z.B. in E-Mail-Clients.

- ***Windows*** ist hier zumindest bei X.509 mit seinem *Cryptographic Service Provider* (CSP) einen Schritt voraus, die Integration in Menüs und Applikation muss jedoch noch als unvollständig bezeichnet werden.
- ***OS X:*** Schlüsselbund mit umfassender Integration in alle Anwendungen
- ***Linux:*** Als systemweiter Zertifikats-Daemon dient zwar wie beschrieben der gnome-keyring-daemon, das native GUI dazu (gnome-keyring-manager) wird allerdings seit geraumer Zeit nicht mehr gewartet und ist in den meisten Distributionen nicht mehr enthalten. Stattdessen wird als GUI das der PGP-Suite *Seahorse* verwendet, nur bietet dieses trotz mehrfacher anderslautenden Ankündigungen nach wie vor keinerlei X.509-Funktionalität. Zwar existiert ein Seahorse-Plugin für den Datei-Manager *Nautilus*, dieses stellt die Funktionen „Signieren" und „Verschlüsseln" aber eben nur für die im Keyring enthaltenen PGP-Zertifikate zur Verfügung.

5. Standard-Anwendungen

5.1. gemeinsame Aspekte

5.1.1. Benutzer- & Token-Zertifikate

Will man den Standard-Anwendungen unter anderen Betriebssystemen als Windows ein auf Chipkarte gespeichertes Zertifikat mit Betriebssystemmitteln (also ohne Verwendung einer gesonderten BKU) zentral zugänglich machen, so hat sich PKCS#11 als die Standard-Schnittstelle der Wahl erwiesen. Da die meisten Applikationen Mozilla NSS verwenden, existieren dzt. sogar zwei Methoden, um das zu erreichen, nämlich:

- Hinzufügen der jeweiligen PKCS#11-Library in den Bereich „*Security Modules*" der Applikation (vgl. z.B. Abb. 5). Dies ist unterschiedlich gut implementiert und muss natürlich für jede Applikation gesondert vorgenommen werden, funktioniert aber im Grunde sehr gut. Eine Vielzahl von Karten wird bereits unterstützt (z.B. durch das OpenSC-Projekt), auch konnte das IT Solution PKCS#11-Modul unter Linux und OS X erfolgreich getestet werden[16].

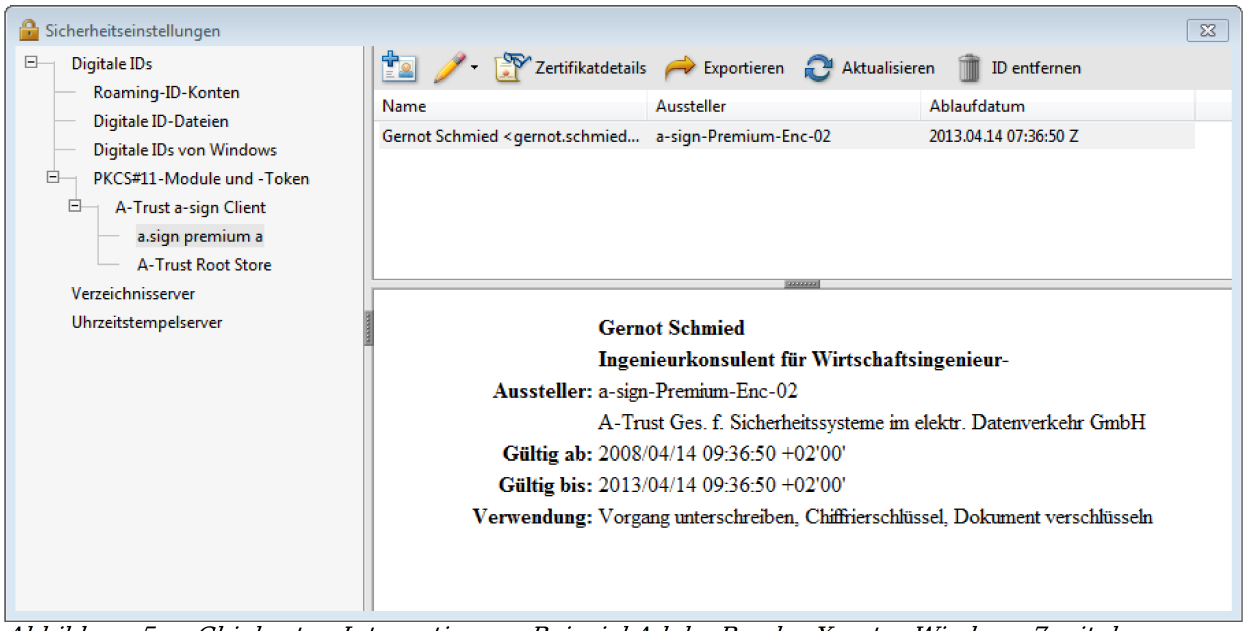

Abbildung 5: Chipkarten-Integration am Beispiel Adobe Reader X unter Windows 7 mit dem PKCS#11-Modul des a.sign Client.

- Zentral lässt sich dieses Interface z.B. unter Linux über den Gnome-Keyring installieren:

16 Da diese Module "nur" eine transparente Schnittstelle zwischen dem PKCS#11-Format und der eigentlichen Chipkarte (auf Basis ihrer herstellerspezifischen APDUs) bereitstellen, wäre es wünschenswert, dass jeder Hersteller für seine Karten ein PKCS#11-Modul anbietet - oder aber, dass man bei Beschaffungsvorhaben grundsätzlich nur Hersteller berücksichtigt, die das tun.

- Via CLI (*ssh-add*) lässt sich in einem ersten Schritt das PKC#11-Modul der ssh-Komponente des gnome-keyring-daemon zugänglich machen (diese verhält sich wie ein Drop-In Replacement für den OpenSSH Client).
- Für die NSS-Applikationen wiederum existiert ein PKCS#11-Modul, das den Zugriff auf alle Funktionen des gnome-keyring-daemon (via seiner PKCS#11-Komponente) bietet und einfach als *Security Module* hinzugefügt wird; letzteres muss zwar wieder in jeder einzelnen Applikation konfiguriert werden, jedoch reduziert sich wenigstens die Komplexität der Installation eines neuen oder geänderten Zertifikatsträgers auf eine einzige (systemweite) Schnittstelle.

Generell ist die SmartCard-Unterstützung der meisten Standard-Applikationen bei weitem noch nicht dort, wo sie im Sinne einer einfachen Handhabung sein müsste, Mozilla Thunderbird stelt hier eine löbliche Ausnahme dar.

Lässt sich auch ein SmartCard-basiertes Zertifikat i.d.R. auf irgendeine Weise der Applikation zugänglich machen, so fehlt es doch oft an der entsprechenden Benutzerfreundlichkeit bei der Verwendung: Z.B. würde man hoffen, dass eine Applikation beim Versuch, ein hinterlegtes SmartCard-Zertifikat zu verwenden, einen entsprechenden Hinweis einblendet, wenn die Karte bzw. der Leser erst an-/eingesteckt werden müssen; stattdessen verweigern die meisten Applikationen einfach den Dienst, und zwar mit mehr oder weniger kryptischen Fehlermeldungen:

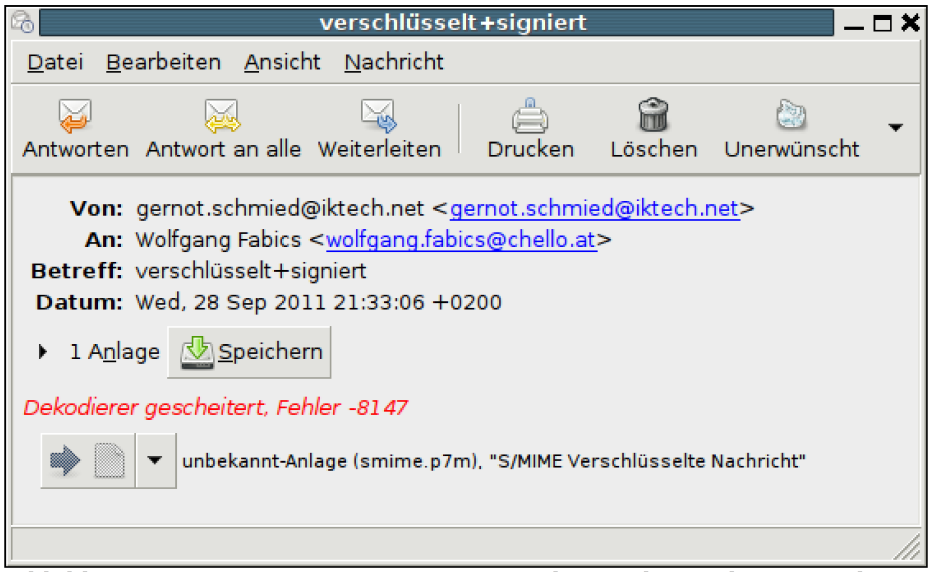

Abbildung 6: Das passiert, wenn man z.B. bei Evolution den Kartenleser nicht angeschlossen hat - allerdings reicht es nicht, das nachzuholen: Evolution muss auch neu gestartet werden!

Ein Hauptproblem bei der Verwendung qualifizierter Zertifikate stellt jedoch nach wie vor die Unterstützung der Zertifikats- bzw. Schlüsselformate durch die Applikationen und das Betriebssystem dar, und zwar in zweierlei Hinsicht:

- ***ECDSA-Unterstützung:*** Durch die Entscheidung, ECDSA viel zu frühzeitig zum Standard-Algorithmus für qualifizierte (Bürgerkarten-)Zertifikate zu machen, wurde deren generische Nutzung

auf Jahre hinaus zuverlässig verhindert. Diese Situation hat sich mittlerweile etwas gebessert, einesteils durch das Rollout von ECDSA-Zertifikaten mit mittlerweile „üblichen“ Schlüsselformaten (NSA Suite B P-256), andererseits durch den immer weiter verbesserten ECDSA-Support in Applikationen und Betriebssystemen.

Tatsächlich werden zumindest die Schlüsselparameter (Algorithmus, Schlüssellänge usw.) bereits vglw. oft korrekt erkannt und dargestellt - bei der konkreten Verwendung von ECDSA-Schlüsseln (auch derjenigen nach NSA Suite B P-256) stürzen etliche der getesteten Standard-Applikationen (ohne spezifische ECDSA-Addons) jedoch nach wie vor einfach ab.

Abbildung 7: ein Blick auf das qualifizierte Zertifikat der e-Card G3 mit dem Mozilla Certificate Viewer: „Could not verify this certificate...“, obwohl (hier nicht dargestellt) der Zertifikatsbaum vollständig ist und der ECDSA-Schlüssel korrekt identifiziert wird

- ***Zertifikats-Profile:*** Ein weiteres Problem bei der Nutzung qualifizierter Zertifikate stellt die Unterstützung der verwendeten Zertifikats-Parameter und -Erweiterungen dar. Insbesondere die rechtlich unbedingt erforderlichen Parameter zur Identifikation als „qualifiziertes Zertifikat“

(*qc-Statements*) werden so gut wie nicht erkannt und führen oft dazu, dass solche Zertifikate entweder nicht importiert werden können[17], dass sie nach dem Import nicht verwendet werden können, oder dass die Applikation bei einem der beiden Versuche einfach abstürzt.

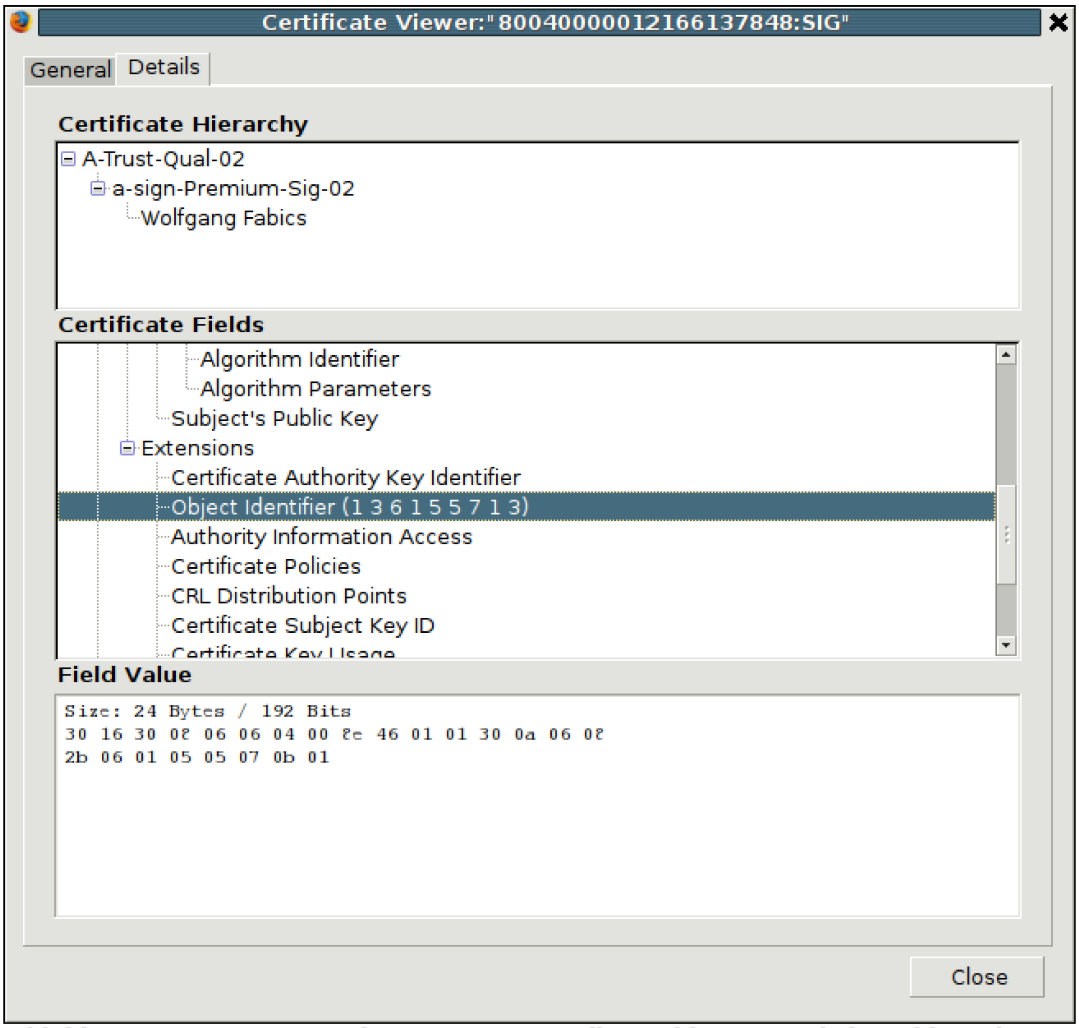

Abbildung 8: Hier zeigt sich erst, womit Mozilla in Abb. 7 eigentlich Probleme hatte: die Certificate Extension mit OID 1.3.6.1.5.5.7.1.3 wird nicht erkannt - und das ist das „qc-Statement“ gem. RFC 3039. Welcher „normale“ Benutzer soll das wohl herausfinden?

Welcher der beiden Aspekte in einem konkreten Fall den Import oder die Verwendung des Zertifikats nun verhindert, ist wiederum selbst für den erfahrenen Benutzer kaum festzustellen: Die entsprechenden Fehlermeldungen sind – wenn sie überhaupt auftauchen – entweder unbrauchbar („Unklassifizierter Fehler in der Befehlsabarbeitung“ oder „Error 5103“) und/oder nicht dokumentiert, aussagekräftige Log-Einträge sucht man meist vergeblich.

17 Gemäß den Standards darf eine Anwendung ein Zertifikat gar nicht importieren, wenn sie ein als „*critical*“ definiertes Attribut nicht interpretieren kann.

Hinzu kommt, dass Fragen nach speziellen österreichischen bzw. europäischen Zertifikatsproblemen in den überwiegend internationalen bzw. US-amerikanischen Support- und Benutzerforen kaum Beachtung finden – ein Grund mehr, warum aus unserer Sicht bei der Entwicklung eines europäischen Signatur(rechts)-Rahmens eher auf die rechtssichere Verankerung weltweit gebräuchlicher und tatsächlich implementierter Verfahren hätte gesetzt werden sollen, anstatt *de facto* eine technisch und rechtlich völlig neue „Signaturwelt“ zu schaffen, für die mangels Relevanz so gut wie keine internationale Unterstützung gegeben ist.

5.1.2. X.509 Root- & Intermediate-Zertifikate

Allen Applikationen ist gemeinsam, dass sie X.509 End-Zertifikate (von Benutzern oder auch anderen Computern) nur dann verwenden können (bzw. dürfen), wenn sie die Zertifikatskette lückenlos bis zum entsprechenden *Root*-Zertifikat (der „Wurzel“ des virtuellen Zertifikatsbaums) nachvollziehen können. Dazu müssen neben dem End-Zertifikat auch das jeweilige *Root*-Zertifikat sowie die in der Zertifikatshierarchie dazwischenliegenden *Intermediate*-(„Zwischeninstanz“-)Zertifikate

- lokal vorhanden,
- installiert und
- mit dem erforderlichen Benutzervertrauen ausgestattet

sein. Dies stellt den Benutzer vor gleich drei Herausforderungen, bevor er mit der Verwendung seines eigenen Zertifikats überhaupt beginnen kann:

- ***Besorgen der Zertifikate:*** Für so gut wie alle namhaften Zertifizierungsstellen (private wie staatliche) wird dies in aller Regel für den Benutzer in der Form erledigt, dass für das jeweilige Betriebssystem ein Installationspaket mit allen weltweit bekanntgegebenen *Root*- (und ggf. *Intermediate*-)Zertifikaten existiert. Diese Pakete (z.B. „Root Certficates“ bei Windows, „ca-certificates“ bei Linux) sind normalerweise bereits vorinstalliert, lassen sich bei Bedarf aber auch einfach über die Paketverwaltung des Systems nachinstallieren. Die Überwachung der Qualität und Gültigkeit der enthaltenen Zertifikate wird von Herausgeber des Pakets (Betriebssystem-Hersteller oder Maintainer) überwacht.

  Mit österreichischen Zertifikaten ist der Benutzer jedoch weitgehend auf sich allein gestellt: Zumindest für Linux gibt es weder von A-Trust noch der RTR sämtliche erforderlichen CA-Zertifikate in einem der großen Zertifikatspakete – was unverständlich erscheint, wenn man zum Vergleich die Liste privater und staatlicher Anbieter betrachtet, die dort sehr wohl vertreten sind. Der Benutzer muss daher von der Website des Zertifikatsanbieters zunächst einmal die benötigten Zertifikate herunterladen. A-Trust bietet verschiedene Installations-Paket an, mit denen diese samthaft per Mausklick im Betriebssystem (Windows) bzw. (zumindest theoretisch) in den Anwendungen Mozilla Firefox/Netscape und Thunderbird (via Extension „atrusttools.xpi“)[18] installiert werden können. Für andere Betriebssysteme und Applikationen ist ein Archiv mit

18 Diese Extension ist aber weder mit aktuellen Versionen von Thunderbird noch mit irgendeiner getesteten Version von Firefox funktionsfähig.

den Zertifikaten herunterzuladen - und danach jedes einzelne zu installieren.

- ***Installation:*** Steht keine der genannten “einfachen” Methoden zur Zertifikatsinstallation zur Verfügung, so muss sich der Benutzer zunächst darüber informieren, in welchem Format sein System bzw. seine Applikation Zertifikate akzeptiert. Für X.509-Zertifikate existiert eine Vielzahl von Dateiformaten, die häufigsten sind PKCS#7 (*.p7b), DER-kodiert (*.crt) und PEM (*.pem)[19]. In der Regel wird dies am schnellsten durch beispielhaftes Ausprobieren zu erreichen sein, was aber auch nicht immer gleich zum Ziel führt[20] – der Benutzerfreundlichkeit ist es jedenfalls kaum zuträglich.

  Ist diese Hürde genommen, so kann man mit dem mühsamen Zertifikats-Import beginnen – da die Mehrfachauswahl von Zertifikatsdateien beim Import in keiner uns bekannten Anwendung möglich ist und wie nachfolgend beschrieben auch jedes einzelne Zertifikat gesondert konfiguriert werden muss. Mühsam finden wir auch die Tatsache, dass ein Zertifikat, dessen Hierarchie nicht lückenlos nach oben geschlossen werden kann, in keiner getesteten Anwendung überhaupt importiert wird – stattdessen muss der Benutzer jedes zu importierende Zertifikat (in einem entsprechenden Viewer) überprüfen, sich das zugehörige Aussteller-Zertifikat notieren und dann auf die Suche nach demselben gehen. Hat der den Zertifikatsbaum solcherart erfolgreich „zu Fuß“ abgeschritten, darf er sich beim Import nun in der umgekehrten Reihenfolge wieder bis zum End-Zertifikat „hinunterhangeln“ – so sehr dies auch der „reinen Lehre“ bei der Zertifikatsverwaltung entsprechen mag, so sind benutzerfreundlichere Ansätze denkbar und wünschenswert.

19 A-Trust stellt ihre *Root*- und *Intermediate*-Zertifikate in diesen Formaten bereit, Benutzerzertifikate aus dem A-Trust-Verzeichnisdienst gibt es allerdings nur als *.crt.

20 Z.B. müssen unter Linux in bestimmten Zusammenhängen DER-kodierte *.crt-Zertifikate in *.cer umbenannt werden, um sie importieren zu können.

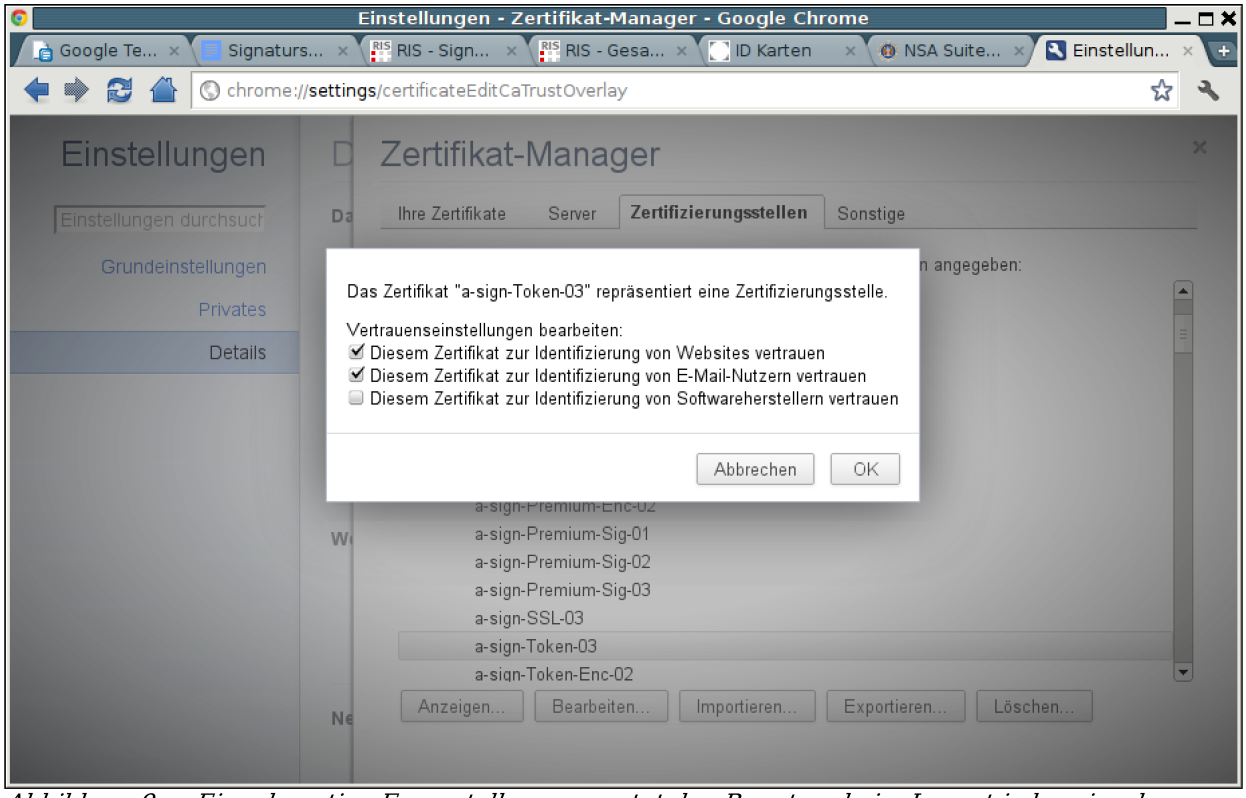

Abbildung 9: Eine derartige Fragestellung erwartet den Benutzer beim Import jedes einzelnen Zertifikats, hier am Beispiel von Google Chrome.

- ***Konfiguration:*** Während des Importvorgangs wird der Benutzer für jedes Zertifikat zur Angabe aufgefordert, für welche Zwecke er dem zu importierenden Zertifikat vertraut[21]. Diese sind (sinngemäß)
  - Identifikation von Servern (Server-Zertifikate)
  - Identifikation von E-Mail-Benutzern (Benutzer-Zertifikate)
  - Identifikation von Software-Herstellern (Herausgeber-Zertifikate)

  Gemäß den hier getroffenen Einstellungen ist ein Zertifikat dann im jeweiligen Bereich des Cert Stores sichtbar und kann für den angegebenen Zweck verwendet werden (die Vertrauensstellung eines CA-Zertifikats wird dabei an die darunterliegenden Zertifikate vererbt).

  Problematisch dabei: Handelt es sich nicht um ein End-Zertifikat, so ist dem Benutzer aus den Zertifikatsangaben alleine nicht ersichtlich, für welche Zwecke das Zertifikat letztlich eingesetzt werden soll: Die o.g. Attribute sind den CA-Zertifikaten fremd, vorhandene X.509-Attribute wie *Basic Constraints* oder *(Extended) Key Usage* behandeln sinngemäß zwar ähnliche, konzeptionell jedoch grundlegend andere Themen.

  Als pragmatischen Ansatz wird der Benutzer also - mangels besseren Wissens - einfach alle drei Zwecke als zulässig auswählen, was oft auch

21 Diese Notwendigkeit liegt grundsätzlich im Konzept der „*trustargs*“ von Mozilla NSS begründet, viele andere Anwendungen und Cert Stores haben dies aber gleichlautend oder sinngemäß übernommen.

funktioniert. Anwendungsabhängig kann dies aber auch dazu führen, dass Zertifikate im „falschen“ Bereich des Cert Stores angesiedelt werden und für den eigentlich angestrebten Zweck dann nicht verwendet werden können – oder aber, dass die Vertrauensstellung des Zertifikats überhaupt nicht berücksichtigt wird[22].

5.2. Web-Browser

Der Web-Browser stellt in seiner Eigenschaft als reine Viewer-Applikation keine Herausforderung bzgl. digitaler Signatur und/oder Verschlüsselung von Dateien dar. Einzige hier relevante Aufgabe ist die Authentifizierung von und bei mittels HTTPS gesicherten Websites – und BKUs (sowohl online als auch lokal). Kernaspekt der *Usability* ist daher in diesem Zusammenhang das Zertifikatsmanagement (Stamm-, Zwischen- und Endzertifikate), das je nach Software durchaus unterschiedlich erledigt wird:

- Die Windows-Browser verfügen allesamt über ein eingebautes Interface zum zentralen Windows CSP, wodurch sich das Zertifikats-Management erheblich vereinfacht.
- Gleiches gilt für Safari und Google Chrome hinsichtlich der Schlüsselbundverwaltung von OS X[23].
- Für Mozilla Firefox& Thunderbird stellt A-Trust eine betriebssystemunabhängige Extension (XPI) bereit, mit der (theoretisch) die Root- und Intermediate-Zertifikate gesammelt installiert werden könnten[24].

In allen anderen Szenarien muss der Benutzer entweder auf ein PKCS#11-Modul hoffen, das die Schnittstelle zu einem gemeinsam genutzten Certificate Store bereitstellt, oder aber alle erforderlichen Zertifikate manuell importieren (vgl. 5.1.).

5.3. E-Mail-Clients & E-Mail-Zustellung

E-Mail-Signatur und -Verschlüsselung mit X.509-Zertifikaten wird grundsätzlich via S/MIME realisiert und funktioniert aufgrund der dzt. so gut wie nicht gegebenen Anwendungsunterstützung qualifizierter ECDSA-Zertifikate eigentlich nur mit RSA-Zertifikaten[25]. Die Bereitstellung der Karten-Zertifikate erfolgt entweder via PKCS#11 Module oder Windows Crypto Service Provider (CSP). Die grundsätzliche Schwierigkeit besteht einzig darin, die involvierten Chipkarten

22 So geschehen unter Linux mit Google Chrome und dem MOCCA-Server-Zertifikat für die lokale BKU. Die hierbei erforderlichen manuellen Eingriffe in die Cert-DB (via CLI) sind dem normalen Benutzer nun wirklich nicht zuzumuten.

23 Firefox verhält sich unter OS X anders und verwendet sein eigenes Cert-Store.

24 Diese Extension ist aber weder mit aktuellen Versionen von Thunderbird noch mit irgendeiner getesteten Version von Firefox funktionsfähig.

25 Die Behandlung von elektronischen Zustelldiensten würde den Rahmen dieser Studie sprengen und ist für die generelle Benutzerwahrnehmung nicht bedeutsam. Solche Dienste (die auf web-basiertem Up-/Download basieren und nicht auf E-Mail) versuchen lediglich, das bei Verwendung „echter“ E-Mail-Protokolle gegebene Versandrisiko zu mindern (→ erhöhte Zustellsicherheit).

anzusprechen bzw. die RSA-Zertifikate auszuwählen und die erforderlichen Root- und Intermediate-Zertifikate zu importieren (vgl. auch Punkt 5.1).

- Der Test mit ***Outlook 2010*** und S/MIME schlug mit RSA-Zertifikaten fehl.
- ***Evolution***, ein weit verbreiteter Linux E-Mail-Client, unterstützt sowohl PGP als auch S/MIME und auch PKCS#11-Module (erfolgreicher Test mit der e-Card G3 und dem PKCS#11-Modul von IT Solution). Evolution verwendet sowohl eine typische NSS cert8.db als auch andere (camel-cert.db, key3.db) und bringt seine Zertifikate in einer eigenen Verzeichnisstruktur mit. Im Zusammenhang mit Signaturen und Verschlüsselung kann das Programm erstaunlich viel erstaunlich gut (z.B. gleichzeitige Anwendung von PGP und S/MIME in einer und derselben E-Mail), jedoch erschließen sich diese Möglichkeiten dem Benutzer nicht ohne weiteres: das GUI zur X.509-Zertifikats- und Kartenverwaltung muss nämlich schlicht als untauglich bezeichnet werden.

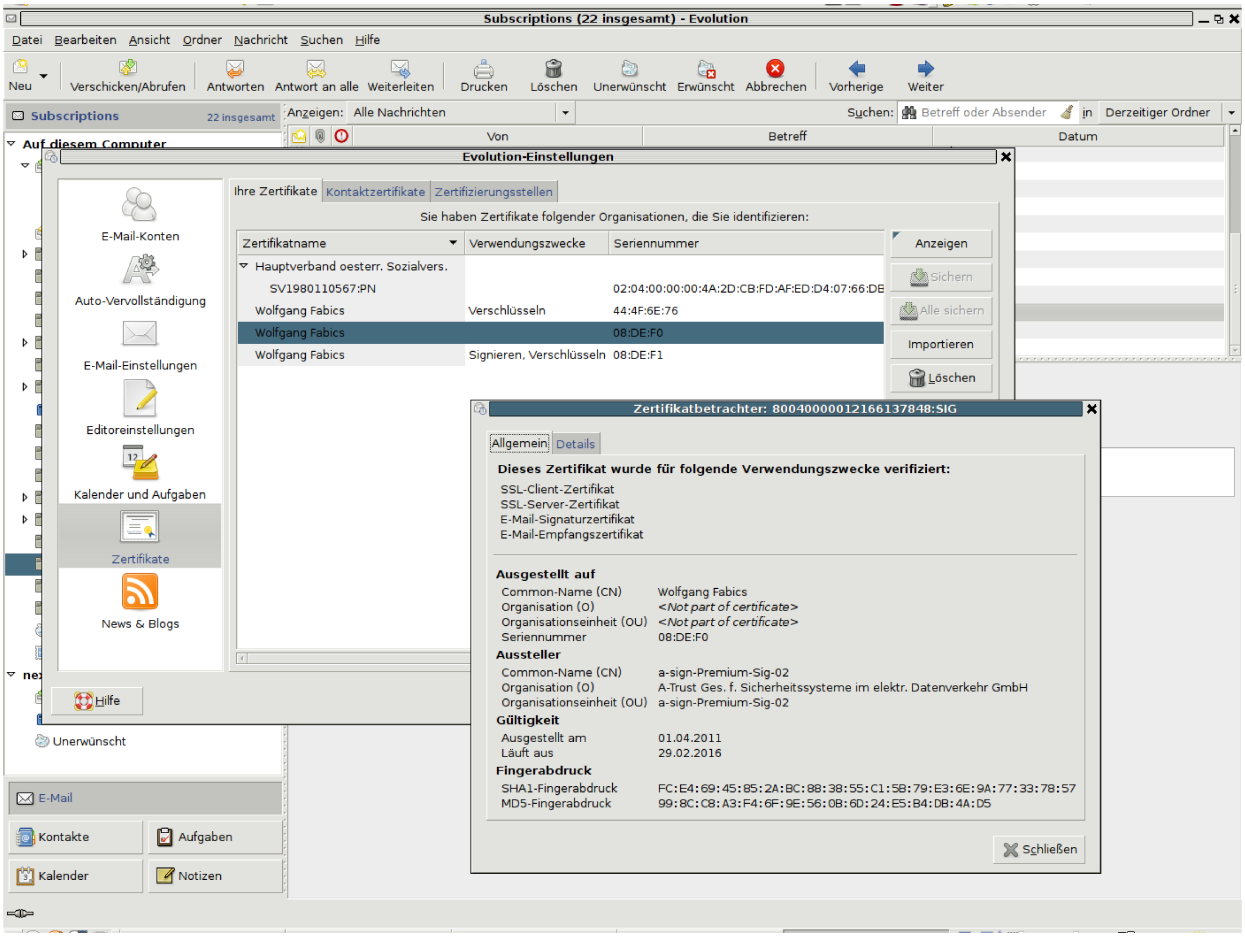

Abbildung 10: Evolution unter Linux sieht das qualifizierte Zertifikat der e-Card G3 (auch hier wieder unter Verwendung der IT Solution PKCS#11 Library) - nur werden weder der ECDSA-Schlüssel noch verschiedene andere Zertifikatsattribute erkannt. Mit dem RSA-Schlüssel auf derselben Karte funktioniert S/MIME jedoch problemlos.

- Für das Einbinden von Security-Modulen wie z.B. PKCS#11 steht keine Funktion zur Verfügung, das muss „zu Fuß“ erledigt werden[26];
- Zertifikate werden nicht nach Format und Verwendungszweck automatisch einsortiert und angezeigt, vielmehr wird je nach Zweck ein bestimmtes Zertifikatsformat erwartet (und ansonsten per default nichts angezeigt);
- der Import von Zertifikaten i.V.m. privaten Schlüsseln (PKCS#12) wird nur unter Abfrage eine Master-Passworts für die NSS-DB gestattet – es gibt aber keine Funktion, dieses Passwort aus Evolution heraus überhaupt zu setzen,
- Stößt Evolution auf Probleme beim Zertifikats-Import, so wird dieses einfach wortlos nicht angezeigt – auf eine Fehlermeldung wartet man vergeblich.

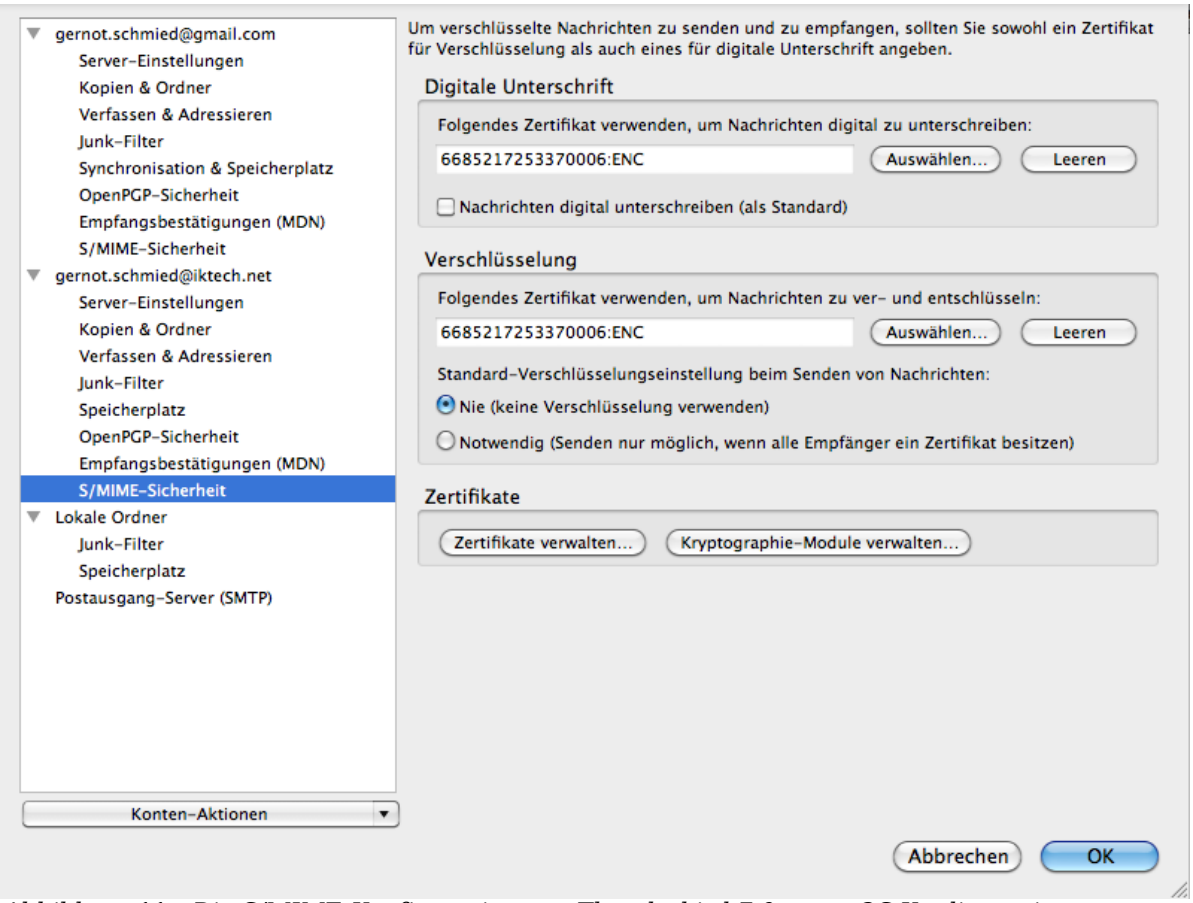

Abbildung 11: Die S/MIME-Konfiguration von Thunderbird 7.0 unter OS X - die „a.sign premium“-Karte wird via PKCS#11-Library von IT Solution angesprochen

- ***Mozilla Thunderbird*** verwendet ebenfalls NSS, speichert diese Informationen jedoch in einer getrennten Datenbankdatei. Die in Firefox

26 Als pragmatischer „Hack“ wurde für die Tests einfach die via Thunderbird-GUI mit dem PKCS#11-Modul vorkonfigurierte cert8.db von Thunderbird in das entsprechende Evolution-Verzeichnis kopiert... für den durchschnittlichen Benutzer jedoch eher wenig zu empfehlen.

importierten Zertifikate und Module stehen somit nicht automatisch auch Thunderbird zur Verfügung – dafür, dass es sich bei den beiden Programmen um zwei Teile einer „Suite“ eines und desselben Anbieters handelt, eigentlich sonderbar. Thunderbird wurde unter Windows 7 mit der PKCS#11-Library von A-Trust sowie unter Linux und OS X mit derjenigen von IT Solution erfolgreich getestet.

- ***Apple Mail***, das OS X Standard E-Mail-Programm, unterstützt grundsätzlich ebenfalls S/MIME, sofern zu einer konfigurierten E-Mail-Adresse im Schlüsselbund ein privater (RSA-)Schlüssel gefunden wird[27].

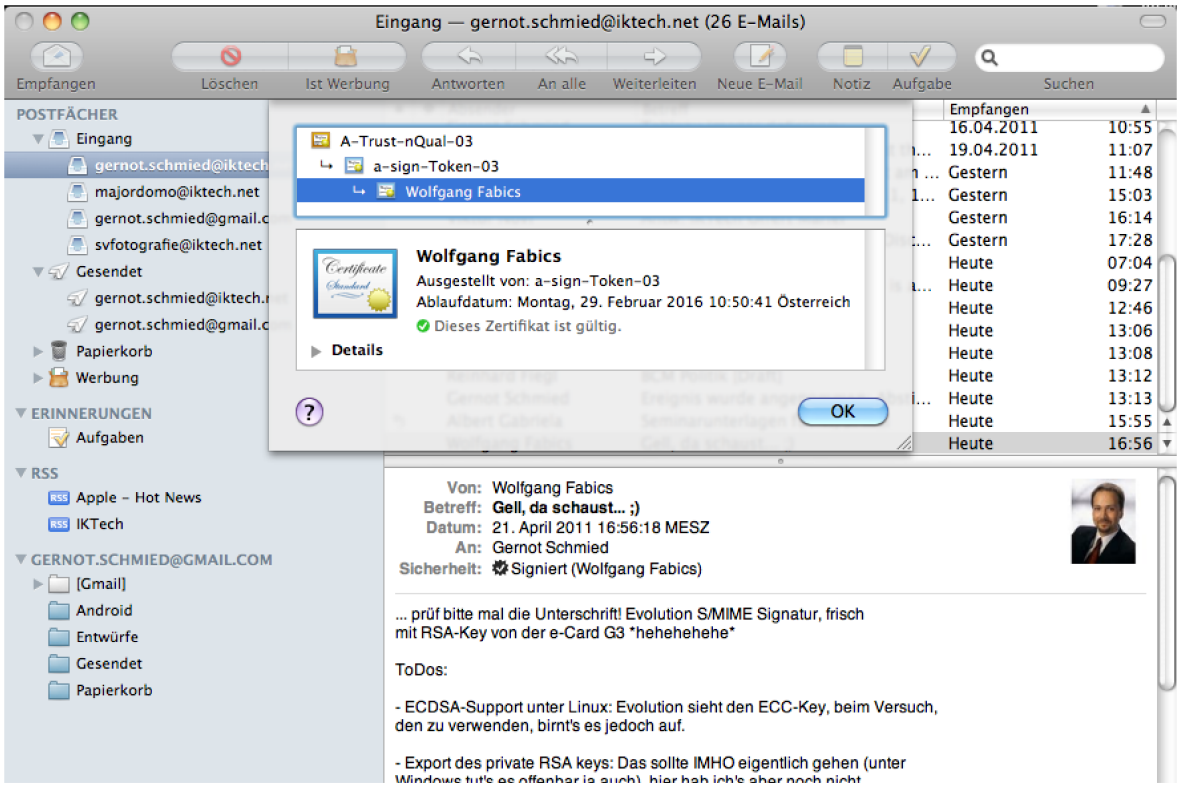

Abbildung 12: mit dem RSA-Zertifikat der e-Card G3 signierte E-Mail von Evolution (via PKCS#11-Library von IT Solution) aus Sicht von Apple Mail beim Empfänger - tadellos!

Alle erwähnten E-Mail-Clients unterstützen PGP entweder nativ (z.B. Evolution, sobald GnuPG am System installiert ist) oder via Plugins/Addons.

5.4. Adobe Reader & Acrobat

Die Adobe-PDF-Suite stellt die wohl meistgenutzte Standard-Umgebung für den Umgang mit PDF-Dateien dar. Es existiert eine Vielzahl anderer Lösungen (meist reine Reader-Applikationen), die jedoch i.d.R. weder in Qualität noch

27 Es hat den Anschein, als ob sich dieses Feature nicht mit installiertem PGP-Module (GPGmail) verträgt. Bei dessen Verwendung sind die Buttons für X.509 S/MIME im E-Mail-Composer nicht verfügbar. Darüber hinaus ist im E-Mail-Programm unter OS X 10.7 „Lion“ in den Account-Settings nun eine explizite Zertifikatsauswahl möglich.

Funktionalität oder Verbreitung an Adobe-Produkte heranreichen und deshalb hier nicht betrachtet werden sollen.

Folgende Schwächen sind aus unserer Sicht vorwiegend für das eingeschränkt positive Benutzererlebnis bei der Verwendung digitaler Signaturverfahren mit Adobe-Programmen maßgeblich:

- uneinheitliche Implementierung des Adobe Readers unter Windows, Apple OS X und Linux: keine Möglichkeit, unter Linux und OS X den LDAP-Server von A-Trust in den Einstellungen anzugeben (kein Eintrag für „Verzeichnisserver“);
- Plugins installieren sich nicht zuverlässig in gleichzeitig installierten Adobe Reader- und Acrobat-Anwendungen – vermutlich wird nur die als Default-Anwendung für PDF hinterlegte Variante berücksichtigt;
- Die Konfiguration der Adobe-Sicherheitseinstellungen ist für Laien in ihrer Bedeutung schwer verständlich (vgl. Abb. 13);
- Adobe kann mit RSA-Zertifikaten nativ umgehen, nicht jedoch mit ECDSA. Dafür sind spezifische Plugins erforderlich.

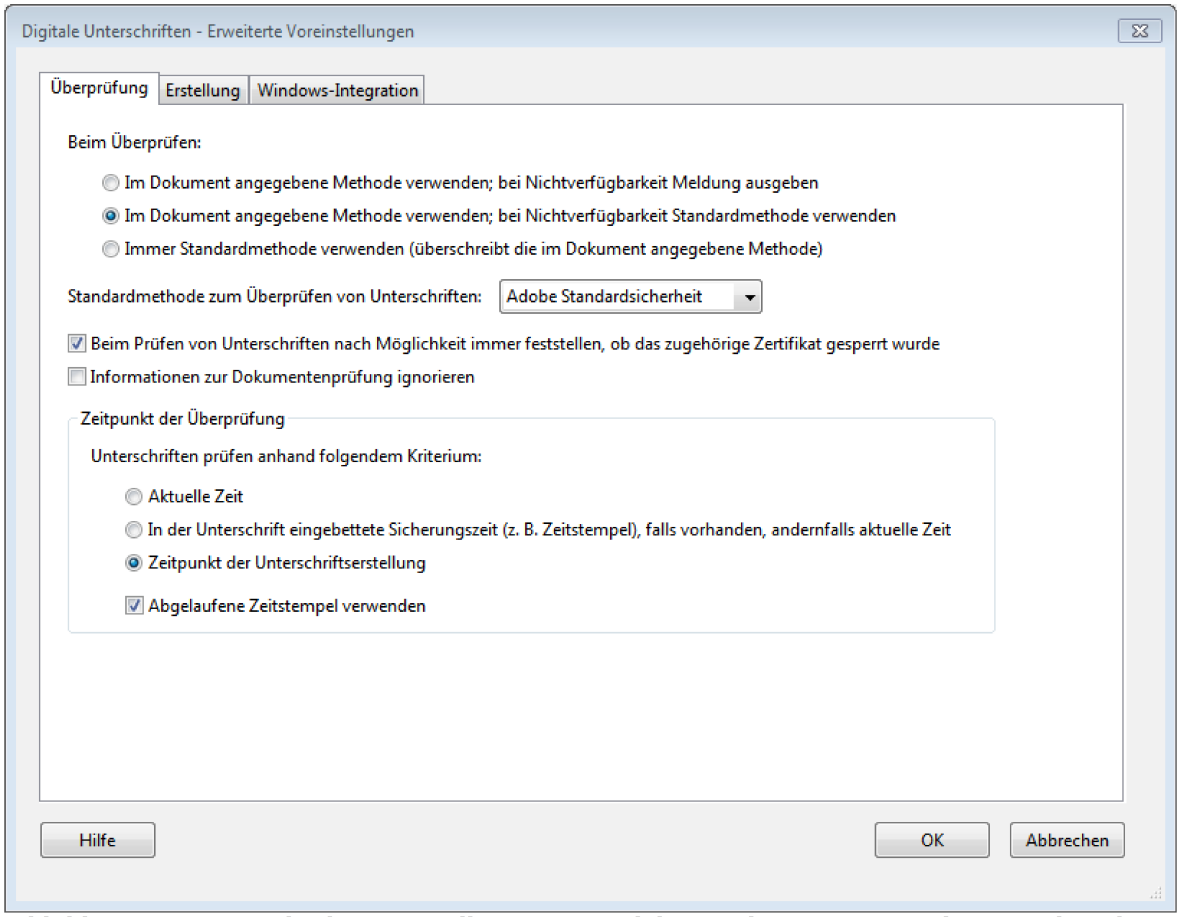

Abbildung 13: Die Sicherheitseinstellungen von Adobe Reader X unter Windows 7. Ohne das Hintergrundwissen, dass hier das Verhalten im Umgang mit der im (unsichtbaren) Signaturblock von PDF-Dokumenten normalerweise verankerten Angabe des „Signature Handlers" (also desjenigen Programms, mit dem eine Unterschrift geprüft werden soll) eingestellt wird, versteht der durchschnittliche Benutzer hier wohl kaum, worum es genau geht. Auch wäre eine Bezeichnung wie „Unterschrift mit Adobe prüfen" wohl aussagekräftiger als „Adobe Standardsicherheit". Zur Prüfung österr. qualifizierter Signaturen muss hier übrigens das (hoffentlich installierte) ECDSA-Plugin von A-Trust bzw. IT Solution anstelle der „Adobe Standardsicherheit" ausgewählt werden.

Es ist nicht davon auszugehen, dass das unter Windows verfügbare PDF-AS-Plugin für Adobe auch zukünftig unterstützt werden wird; umso mehr erscheint das funktionierende Interworking zwischen Adobe mit seinem weltweit genutzten Signaturformat und den österr. Chipkarten (via PKCS#11) nötig und wünschenswert.

5.5. Office-Suiten

OpenOffice, LibreOffice:

Das ggw. verfügbare PDF-AS-Plugin von A-SIT für OpenOffice funktioniert gut unter aktuellen Installationen von OpenOffice und LibreOffice. Es wäre wünschenswert, dass dieses Plugin auch weiterhin gewartet wird, da LibreOffice

für alle Plattformen frei verfügbar ist, das Plugin generisch einsetzbar ist und viel intuitiver als PDF-Over funktioniert.

Die Interaktion mit Chipkarten und Zertifkaten via Standard-Methoden (PKCS#11, Cert-Stores) erfolgt unter Linux lt. Dokumentation via NSS-DB von Thunderbird – falls dieser nicht installiert ist, über diejenige von Firefox. Entsprechende Tests waren aber noch nicht durchschlagend erfolgreich, die Situation hat sich in den aktuellsten Versionen evtl. auch geändert[28]. Die Chipkarten-Zertifikate sind zwar sichtbar, eine ODT-Dokumenten-Signatur ließ sich bislang jedoch aus ungeklärten Gründen nicht erstellen (keine Fehlermeldung, kein Log, keine Dokumentation).

MS Office Word:

Das PDF-AS-Plugin für MS Office Word ist veraltet, funktioniert in Office 2010 nicht mehr und führt zu Abstürzen.

Auf der A-Trust-Website wird unter „Zertifikate in Standardanwendungen nutzen“ der Eindruck einer umfangreichen Unterstützung suggeriert – diese könnte auch gemessen an der von A-Trust bereit gestellten substanziell veralteten Dokumentation von der traurigen aktuellen Realität nicht weiter entfernt sein. Selbst für Fachleute ist dies ein Forschungsprojekt, falls es überhaupt mit aktuellen Anwendungen funktioniert.

5.6. TrueCrypt

TrueCrypt hat sich als hochwertige und freie Verschlüsselungslösung zum *„de facto“*-Standard entwickelt und funktioniert unter Windows und Linux mit PKCS#11 und RSA-Zertifikaten ausgezeichnet. Der Test unter OS X mit der PKCS#11-Library von IT Solution schlug leider fehl.

28 Diese Aussage beruht auf Tests mit OOo 3.2 unter Ubuntu 8.04. Leider verhindert zum Zeitpunkt der Erstellung der Studie ein ungelöstes Problem mit dem Kartenleser-Treiber die Tests auf einer aktuelleren Installation.

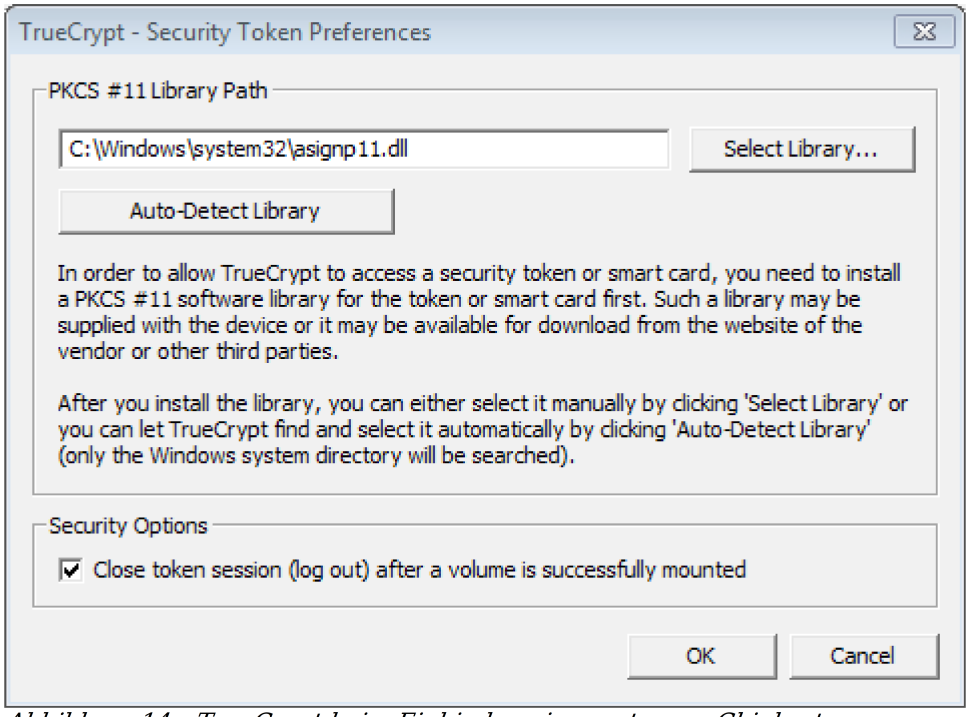

Abbildung 14: TrueCrypt beim Einbinden einer externen Chipkarten-Schnittstelle via PKCS#11 (hier am Beispiel der A-Trust Library des a.sign Client unter Windows 7)

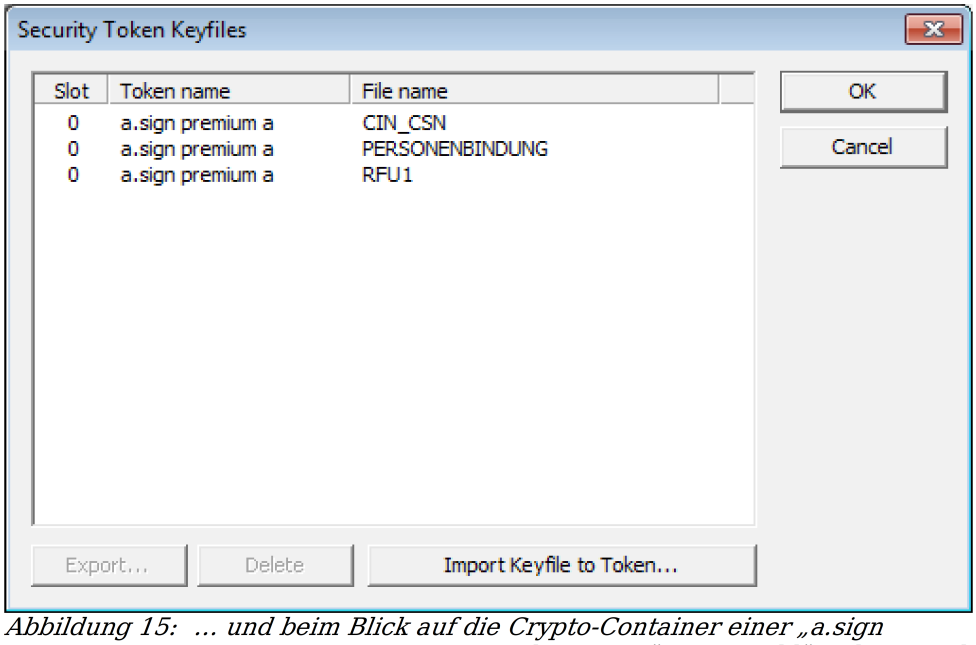

Abbildung 15: ... und beim Blick auf die Crypto-Container einer „a.sign Premium"-Karte. Datei- und Datenträger-Verschlüsselung auch mit österr. Chipkarten ganz einfach – so nützlich ist PKCS#11.

5.7. Handy-Signatur (Google Android & Apple IOS)

Für ***Google Android*** ist die Applikation sPDF mit mobiler Signatur im *Android Market* verfügbar und erscheint vielversprechend (http://langkamp-it.com/), eine volle Produktiv-Version wird in Kürze einsetzbar sein und kann grundsätzlich empfohlen werden.

Für ***Apple IOS*** ist uns ggw. keine mobile Signatur-App bekannt. Eine entsprechende Entwicklungsaufgabe könnte unserer Ansicht nach ein ideales Thema für eine Diplomarbeit darstellen - oder für ein netIdee-Projekt ☺.

Abbildung 16:
die Oberfläche von sPDF auf dem HTC Android - mehr braucht man als Benutzer wirklich nicht, um ein PDF via Mobilsignatur zu unterschreiben; das Signaturformat ist übrigens auch hier PDF-AS.
(Quelle: buergerkarte.at)

6. Erkenntnisse & Vorschläge

6.1. systemübergreifende Aspekte

6.1.1. Bürgerkartenumgebungen, Adobe & Co.

An einer BKU führt ggw. im Umgang mit ECDSA kein Weg vorbei, wie auch immer diese angesprochen wird (native Anwendungen a la a.sign oder IT Solution Client oder via Adobe Plugin). ECDSA erfordert (aufgrund der dzt. nicht gegebenen internationalen Unterstützung) zwingend österr. Speziallösungen.

An Adobe Reader/Acrobat führt als *de facto* PDF-Standardanwendung ebenfalls kein Weg vorbei, dies ist auch nicht dispositiv[29].

Die folgen Adobe-Ansätze sind daher aus unserer Sicht parallel erforderlich und zweckmäßig:

- ***qualifizierte*** Adobe-Signatur via ECDSA-Plugin
- ***fortgeschrittene*** Adobe-Signatur via nativem Adobe RSA-Support (CSP, PKCS#11)

Wir sprechen uns für eine kritische Neuevaluierung von PDF-AS als Signaturformat aus, da gegenüber der Adobe-Signatur keinerlei Vorteile (mit Ausnahme des obsoleten Anwendungsfalls „Verifizieren durch Abtippen“) und etliche Nachteile (nicht international genormt, nur online verifizierbar, von Adobe nicht verarbeitbar) zu Buche stehen. Zu berücksichtigen wären dabei insbesondere auch Ansätze zur Unterstützung nativer Adobe-Signaturen durch BKU-nahe Anwendungen/Plugins/Apps[30] (z.B. für die Handy-Signatur in Zusammenhang mit Smartphones/Tablets).

Als „Gesamtlösung“ bliebe die Möglichkeit, auch in Österreich qualifizierte Zertifikate auf Basis von RSA anzubieten, dies würde die Schnittstelle zu und die Nutzung der qualifizierten Signatur in Standard-Anwendungen (wie Adobe) vereinheitlichen und wesentlich erleichtern. Die Notwendigkeit für BKUs bleibt in Österreich dennoch zur Kommunikation mit der Verwaltung und anderen entsprechend abgesicherten öffentlichen Einrichtungen (Stichwort: Personenbindung) bestehen.

Dringend wird empfohlen, hinsichtlich PKCS#11-Libraries eine Zusammenarbeit mit den Projekten „TokenD“ (für OS X) und „OpenSC“ anzustreben.

6.1.2. Ende der „Informations-Schnitzeljagd“

- „Ausmisten“ und Aktualisieren der Inhalte auf www.buergerkarte.at,

29 Es ist wichtig sicherzustellen, dass solche Plugins immer für Acrobat *und* Reader installiert/updated werden und diese Plugins bei Erscheinen einer neuen Major Release auch weiterhin funktionieren und unmittelbar nach Markteinführung verfügbar sind.

30 Grundsätzlich bedarf es auf Smartphones und Tablets bei Verwendung der online-Variante von PDF-AS (https://www.handy-signatur.at) jedoch lediglich eines Browsers und der mobilen Signatur.

- rigoroses Entfernen in aktuellen Umgebungen nicht mehr unterstützter, eingestellter oder nicht mehr funktionierender Software (Office Plugins etc.),
- Zusammenführen bzw. auffindbares und konsistentes Verlinken von bzw. zwischen Inhalten von BKA, RTR, EGIZ, A-SIT und www.chipkarte.at.

6.1.3. Umgang mit Anwendern

„Der Wurm muss nicht dem Angler, sondern dem Fisch schmecken.“

Entwickler bauen *de facto* immer noch Anwendungen für Entwickler (und deren eigenen Debugging-Zwecke), sollten jedoch Endbenutzer ansprechen. Hier ist ein Wechsel des *Mindsets* erforderlich. Dies ist eng verknüpft mit der Frage der Qualitätssicherung. Dringend nötig ist intuitive und robuste Software (Ursachenbekämpfung) anstatt Zeit in aufwändige Benutzerhandbücher und verstärktes Support-Aufkommen zu investieren (Symptombekämpfung). Software muss selbsterklärend, ohne Handbuch installierbar und erlernbar sein und somit das Benutzerverhalten antizipieren.

Benutzer sind in deren „*Comfort Zone*“ abzuholen und angesichts des unbefriedigenden Ist-Zustandes mit Verständnis zu behandeln; auch Signalwirkungen wie z.B. die einer kostenpflichtigen A-Trust Hotline sind hier zu beachten. Es darf an dieser Stelle an die Alterspyramide erinnert werden und somit an die wachsende Menge älterer Mitbürger, die damit ebenfalls zurecht kommen müssen und keine „*Digital Natives*“ sind.

6.1.4. zum Thema Zeitstempeldienste

Der BEV-Zeitstempeldienst sollten im Sinne der Verbreitung der e-Signatur kostenlos sein. Die „Kleingeldsammlerei” für Zeitstempelkontingente ist kontraproduktiv und das Tragen der vglw. geringen Betriebskosten durch die Republik als Service am Bürger wohl argumentierbar.

Die gesamte „*Standalone*“-Anwendungslandschaft (Werkzeuge von A-Trust und IT Solution, ggf. PDF-Over) sollte dringend um das Feature „Zeitstempel“ erweitert werden, in Adobe-Produkten ist dies bereits möglich und funktioniert mit dem QZSD des BEV auch zufriedenstellend.

Zeitstempelung während des Signaturvorgangs sollte zu einer Selbstverständlichkeit werden und bedarf auch keiner Benutzer-Interaktion. Wiederum ist hier der eidgenössische *LocalSigner* Client als geradezu mustergültige Umsetzung zu erwähnen.

6.2. Empfehlungen an Anwender

Werfen Sie einen Blick auf den Schweizer LocalSigner Client mit einer der vorgestellten PKCS#11-Libraries, es lohnt sich!

Verwenden Sie TrueCrypt mit einer der vorgestellten PKCS#11-Libraries (klappt auch wunderbar im Zusammenhang mit Cloud-Storage wie dropbox)

Verwenden Sie für E-Mail Thunderbird mit einer der vorgestellten PKCS#11-Libraries.

Sollten Sie sich ungern mit S/MIME herumärgern, verwenden Sie PGP-Zertifikate zur E-Mail-Transportsicherung und X.509-Zertifikate ausschließlich für PDF-Attachments.

Vermeiden Sie bei E-Mails, die Sie aus Vertraulichkeitsgründen verschlüsseln, jedenfalls vertrauliche Informationen in der Betreff-Zeile (da diese nicht von der Verschlüsselung umfasst ist), bei Nachrichten mit nur verschlüsseltem Attachment auch im Nachrichtentext.

Verwenden Sie unter Linux und Apple OS X MOCCA als BKU.

7. Zusammenfassung & Ausblick

Die Zukunft und vielleicht auch letzte Verbreitungschance der qualifizierten wie auch der fortgeschrittenen elektronischen Signatur in Österreich liegt nach unserer Überzeugung in einem ergänzenden Nebeneinander der Unterstützung von e-Card als Verbreitungsinstrument und mobiler (Handy-)Signatur. Diese Kombination eröffnet auch Nutzungsmöglichkeiten auf Smartphones und Tablets und ist damit zukunftssicher.

Ein von zumindest A-Trust bereits seit geraumer Zeit angebotener „Workaround“ für die Verwendung von Signaturen in Standard-Anwendungen waren zusätzliche RSA-Zertifikate ergänzend zu den ECDSA-Zertifikaten; diese Option wurde mit der e-Card leider erst in deren letzten Generation nun endlich bereitgestellt. Dies bestätigt wohl implizit unsere Ansichten zur ECDSA-Entscheidung und zum Bedarf an Zertifikaten, mit denen die gängigen Betriebssysteme und Anwendungen auch etwas anfangen können[31].

Der für viele Benutzer nach wie vor unbefriedigende Gesamteindruck ist aus unserer Sicht durchaus auch als Konsequenz nachhaltiger Fehleinschätzungen der Verbreitung von Linux und Apple OS X zu sehen; allmähliche Verbesserungen in dieser Hinsicht scheinen sich jedoch abzuzeichnen.

Wir hegen die Hoffnung, dass diese Studie einerseits nachdenklich stimmen, aber auch Anregungen zum Beschreiten neuer Wege unter Einbeziehung aller interessierten Kreise liefern kann.

31 In diesem Zusammenhang darf nochmals auf die Situation in der Schweiz hingewiesen werden – dort sind qualifizierte Signaturen auf Basis von 2048bit-RSA-Schlüsseln üblich, die von den drei betrachteten Betriebssystemen und deren Applikationen problemlos unterstützt werden. Nach unserem Verständnis des Anhangs zur SigV wäre dies auch in Österreich rechtlich zulässig.

8. Quellenverzeichnis & Links

- http://www.buergerkarte.at/
  Ressourcen rund um die Bürgerkarte
- http://jce.iaik.tugraz.at/
  Stiftung „*Secure Information and Communication Technologies*“ (SIC)
- http://www.a-sit.at/
  Zentrum für sichere Informationstechnologie Austria, A-SIT
- http://www.egiz.gv.at/
  E-Government-Innovationszentrum
- http://egovlabs.gv.at/
  OpenSource-Plattform des „Digitalen Österreich“
- http://www.digitales.oesterreich.gv.at/
  Die Plattform „Digitales Österreich“ (PDÖ) ist das Koordinations- und Strategie-Gremium der Bundesregierung für E-Government.
- http://www.chipkarte.at, http://www.svc.co.at/
  Sozialversicherungs-Chipkarten Betriebs- und Errichtungsgesellschaft m.b.H. (SVC) - rund um die e-Card
- http://www.a-trust.at/
  A-Trust Gesellschaft für Sicherheitssysteme im elektronischen Datenverkehr GmbH
- http://www.itsolution.at
  IT Solution GmbH
- http://www.handy-signatur.at
  rund um die mobile Signatur
- http://signatur.rtr.at/
  RTR/TKK
- http://bka.gv.at
  Bundeskanzleramt
- http://www.e-service.admin.ch/wiki/display/openegovdoc/LocalSigner
  Open eGov LocalSigner der Schweizerischen Eidgenossenschaft
- http://www.xyzmo.com
  Pen & Signaturlösungen
- http://langkamp-it.com/
  die Heimat der Android-App sPDF
- http://smartcardservices.macosforge.org/trac/wiki/tokend
  OS X TokenD Project
- http://www.opensc-project.org/opensc
  Smartcard Tools und Libraries
- http://www.gpg4win.org/
  Gpg4win (*GNU Privacy Guard for Windows*) ist ein Kryptografie-Werkzeugpaket für Windows.

- http://www.implbits.com/hashtab.aspx
  Hashtab Tool für Datei-Prüfsummen
- http://www.cryptool.org/
  E-Learning-Programm für kryptographische Verfahren
- http://www.bev.gv.at
  Bundesamt für Eich- und Vermessungswesen (BEV), qualifizierter Zeitstempeldienst (QZSD)
- http://www.schneier.com/cryptography.html
  Sammlung interessanter Arbeiten zu kryptographischen Themen von Bruce Schneier, einem Urgestein der „Crypto-Szene“
- http://www.nsa.gov/ia/programs/suiteb_cryptography/
  Website der NSA mit einem Überblick der Arbeiten und Standards zur vielzitierten „Suite B“

9. Glossar

ACOS	*Austria Card Operating System*, ein Chipkarten-Betriebssystem des österr. Herstellers Austria Card, das z.B. auf den A-Trust-Karten eingesetzt wird
APDU	*Application Protocol Data Unit*, ein Befehls- bzw. Meldungswort im Kommunikationsprotokoll von Chipkarten
API	*Application Programming Interface*, eine Programmierschnittstelle zur Kommunikation mit Anwendungsprogrammen
AppArmor	ein Sicherheits-Framework für Linux, das Anwendungsprogrammen via vorgefertigter Profile den Zugriff auf bestimmte Funktionen und Ressourcen ermöglicht (oder verbietet)
BKU	BürgerKartenUmgebung – stellt via Schnittstelle „Security Layer“ die Kommunikation zwischen Chipkartenfunktionen und Anwendungen sicher
CDSA	*Common Data Security Architecture*, ein plattformunabhängiges Sicherheits-Framework, bisher praktisch nur von Apple eingesetzt
CLI	*Command Line Interface*, Texteingabeschnittstelle; bei Linux via Shell realisiert, bei Windows als “DOS-Fenster” bekannt
CSP	Microsoft *Cryptographic Service Provider*, die zentrale Zertifikats-Schnittstelle unter Windows
ECDSA	„*Elliptic Curve Digital Signature*“-Algorithmus
MOA	*Module Online Application*
NSA	*National Security Agency,* der militärische Nachrichtendienst der Vereinigten Staaten
NSS	*Network Security Services*, ein zertifikatsbasiertes Sicherheits-Framework der Mozilla Foundation
PAM	*Pluggable Authentication Module*; ein modulares Sicherheits-Framework für Linux
PC/SC	*Personal Computer/Smart Card*, verbreiteter

	Gerätetreiber-Standard für den Zugriff auf Chipkarten, unter Linux und OS X als pcscd ausgeprägt
PDF	*Portable Document Format*, eine Suite genormter Dateiformate für Dokumente, ursprünglich von Adobe entwickelt
PDF-AS	spezifisch österr. Signaturformat für die Anwendung in der Verwaltung („→*PDF*-Amtssignatur“)
PGP	*Pretty Good Privacy*, eine verbreitete Alternative zu →*X.509*-Zertifikaten, vornehmlich für die Verschlüsselung & Signatur von E-Mails entwickelt
PIN/PUK	numerische Codes für SIM-Karten-Zugriff/-Entsperrung („*Personal Identification Number*“, „*Personal Unblocking Key*“)
PKCS	*Public Key Cryptography Standards*, definieren u.A. eine →*API*, die eine generische Schnittstelle zu kryptografischen →*Token* spezifiziert (PKCS#11), ein Signaturformat (PKCS#7) und ein Container-Format für die Speicherung von Schlüsselmaterial (PKCS#12)
QZSD	qualifizierter Zeitstempeldienst
RSA	„Rivest-Shamir-Adleman“, ein nach seinen Entwicklern benanntes asymmetrisches Crypto-Verfahren
S/MIME	*Secure Multipurpose Internet Mail Extension*, ein standardisiertes Verfahren für die Verschlüsselung und Signatur von E-Mails auf Basis von →*X.509*-Zertifikaten
SSCD	*Secure Signature Creation Device*, dt. „sichere Signaturerstellungseinheit“, einfach gesagt ein →*Token* (z.B. Chipkarte), der den Anforderungen zur Erstellung qualifizierter Signaturen genügt
STARCOS	ein Chipkarten-Betriebssystem (*Card Operating System*, COS) des dt. Herstellers Giesecke & Devriant das z.B. auf den österr. e-Cards eingesetzt wird
Token	konkret „*Security Token*“, eine Hardware-Komponente, die Verschlüsselungs- und/oder Authentifizierungsinformationen (z.B.

	öffentliche/private Schlüssel, Zertifikate) beherbergt (z.B. USB-Stick, Chipkarte, TPM-Modul) und ggf. auch zusätzliche Sicherheitsfunktionen (→*PIN*, Zugriffssteuerung, Schlüsselerstellung usw.) bereitstellt
X.509	Gruppe von Standards für Zertifikate, Teil der X.500-Suite der ITU
XPI	*Cross-Platform Install*, ein betriebssystemunabhängiges Dateiformat für Installationsdateien der Mozilla Foundation, mit denen Programme wie Firefox oder Thunderbird mit Zusatzfunktionen (*Extensions*) ausgerüstet werden können