Privacy beyond X.509: Sichere Emails, Versuche mit X.509

Gilbert Brands, Bernd Roellgen

Version 1.0, Juli 2016

Rahmenbedingungen

Ausgehend vom Konzept der elektronischen Identität (EI) wird untersucht, wie sich eine Verschlüsselung von E-Mails auf der Basis X.509 – S/MIME durchführen lässt. Bei einer elektronischen Identität (EI) gibt gibt es, im Gegensatz zu X.509 – S/MIME, keine zentrale Stelle, die die Teilnehmer authentifiziert. Eine zentrale Authentifizierung ist im E-Mailbereich auch deutlich weniger interessant als im Serverbereich, da die meisten E-Mails zwischen bekannten Konten ausgetauscht werden und sich eine zusätzliche Authentifizierung dadurch erledigt (an der Kasse im Supermarkt genügt ja auch die Unternehmenskleidung mit dem Namensschild am Revers, ohne dass nach dem Personalausweis gefragt wird).

Viele Standard-E-Mailanwendungen unterstützen in der Regel X.509 von Haus aus. In der ersten Versuchsreihe werden Standard-E-Mailanwendungen ohne zusätzliche, speziell für die EI-Nutzung programmierte Erweiterungen. Es scheint technisch möglich, ohne eine einzige zusätzliche Zeile schreiben zu müssen, Code X.509 derart zu verwenden, dass das sinngemäß eher mit PGP verwandte EI-Konzept verwendet werden kann, ohne dass Standardapplikationen auf irgendeine Weise erweitert werden müssten. Lässt sich das mit einer einfachen Konfiguration und ohne zusätzlichen Bedienaufwand während der Nutzung realisieren, besteht eine Möglichkeit, auch den normalen E-Mail-Nutzer zur Verschlüsselung zu bewegen.

Eine solche (Zwischen-)Lösung wäre auch sicherer: neben der Vermeidung des höheren Aufwands bei der Beschaffung von offiziellen X.509-Zertifikaten und der häufig damit verbundenen Kosten, was normale Nutzer bereits von dieser Verschlüsselungsmöglichkeit abhält, schließt das EI-System auch die Möglichkeit aus, dass private Teile der Schlüssel bei den Agenturen verbleiben und ein Mitlesen der Nachrichten trotz Verschlüsselung ermöglichen.

Eine solche Lösung wäre aber nur ein Zwischenschritt, da das X.509-Verfahren modifiziert wird, ohne das EI-System bereits wirklich einzuführen. Im zweiten Schritt wäre ein zusätzliches EI-Management, das über die hier zunächst verwendete X.509-Zertifikatsverwaltung hinausgeht, einzuführen. Hierbei sind anwendungsspezifische Add-Ons so zu implementieren, dass für Nutzer ohne spezielle Kenntnisse eine einfache und sichere Verwendung resultiert.

Das Ziel einer Entwicklung ist, die Verschlüsselung immer zu gewährleisten, ohne dass Anwender überhaupt etwas davon bei der Nutzung bemerken. „Nicht bemerken“ beschränkt sich allerdings auf die Bedienung, denn die Grundphilosophie der EI besteht darin, dem Nutzer eine deutlich aktivere Rolle beim Schutz seiner Sicherheit zukommen zu lassen. Netzwerksicherheit soll damit, wie die Nutzung von Kraftfahrzeugen und technischen Geräten, ein Bereich der Eigenverantwortung der Nutzer werden.

Die Motivation für den hier beschriebenen Versuch ergibt sich aus der Erkenntnis, dass X.509, ursprünglich zum sicheren Websurfen konzipiert (https://), verfahrensbedingt bislang nur eingeschränkt für E-Mail, Chat, etc. verwendbar ist. Nahezu alle E-Mail-Agenten verfügen jedoch erfahrungsgemäß über X.509-Unterstützung, weil es durchaus Unternehmen gibt, die betriebsinternen E-Mailverkehr damit verschlüsseln. So praktisch X.509 für die Absicherung der innerbetrieblichen Kommunikation ist, so perfekt hat das Verfahren Dank der Beteiligung von Zertifikat-Agenturen zur Kommunikation im Klartext in 99% aller anderen Fälle gesorgt. Die verbreitete Nutzung

von E-Mails und die logische Lücke, zwar über SSL-gesichterte Webseiten in Shops einzukaufen, aber nachfolgend wesentliche Informationen völlig unverschlüsselt über eine E-Mail „bestätigt“ zu bekommen, ist der ideale Einstiegspunkt, das EI-Konzept schrittweise einzuführen.

Zertifikatsoftware

In diesem Versuch wird das X.509-Schema so genutzt, wie es ist, d.h. es werden X.509-Zertifikate erzeugt und in EI-naher Weise eingesetzt, ohne dass bereits ein echtes EI-System daraus entsteht. Für die Erstellung der Zertifikate wird das Programm XCA eingesetzt. XCA ist eine menügeführte OpenSource-Software zum Erzeugen und Verwalten von X.509-Zertifikaten (https://sourceforge.net/projects/xca/) von Christian Hohnstädt und in C++ programmiert. In Konfigurationsdateien sind weitere Datenfelder über den X.509-Standard hinaus definierbar. Für spezielle Felder für das spätere EI-Management können Object Identifier (OID) definiert und eingebaut werden. Für solche OID werden in der Versuchsphase freie Bezeichner im ISO-OID-Baum (http://www.oid-info.com/) verwendet, ohne deren Nutzung allerdings vorläufig zu beantragen.

Durch Menüführung, Vorlagendefinition und übersichtliche Verwaltung der notwendigen X.509-Zertifikate scheint XCA gut geeignet, in der Einführungsphase auch Nichtspezialisten einen Umgang mit EI zu ermöglichen. Versuche haben gezeigt, dass die Schritte von der Erzeugung bis zur Installation der notwendigen X.509-Zertifikate nur wenige Handgriffe erfordern und nicht komplizierter sind als die Einrichtung eines E-Mail-Programms.

E-Mail-Software

Der Teilnehmer am EI-Konzept wird (und soll) seine gewohnte E-Mail-Anwendung weiter nutzen. Für die Versuche bedeutet das allerdings, dass auch verschiedene E-Mail-Anwendungen untersucht werden müssen. Unsere Versuche wurden mit Thunderbird/Linux durchgeführt und sukzessive auf Mail/Windows und Mailsysteme auf Tablets ausgedehnt. Die wesentliche Einschränkung dieser Versuchsserie besteht in der Beschränkung auf gerätenbasierender Software. Webmailanwendungen, die branchenspezifisch zwischen 30% und 50% ausmachen (http://de.statista.com/statistik/daten/studie/210656/umfrage/anteil-der-nutzung-von-webmail-in-verschiedenen-branchen/), sind auf die Unterstützung der Provider angewiesen, d.h. ein wesentlicher Teil der potentiellen Nutzer könnte selbst bei Interesse (zunächst) nicht oder nur durch Umstieg auf eine andere E-Mailverwaltung teilnehmen.

Allgemeines zu Zertifikaten

Wir gehen davon aus, dass der Leser mit den Grundprinzipien der X.509-Zertifikatwesens vertraut ist. Was in der Theorie logisch und einfach aussieht, erweist sich in der Praxis allerdings alles andere als simpel. Das haben auch die Software-Hersteller und Agenturen erkannt, und Versuche, die Sicherheitsversprechen zu retten, führen inzwischen zu einem für den normalen Nutzer (und vermutlich auch für manchen Experten) nahezu undurchschaubaren Wirrwarr an Regeln und Systemverhalten.

Für Verschlüsselung eingesetzte X.509-Zertifikate sind nach den allgemeinen Vorgaben durch Root-Zertifikate zu signieren, deren Einsatzzweck sich auf diese Aufgabe beschränkt. Grundsätzlich sollten von den Anwendungen aber auch Zertifikate, die nicht durch eine der Root-CAs oder selbstsigniert sind, trotzdem annehmbar sein, indem dem Nutzer das Zertifikat präsentiert und ihm eingeräumt wird, eine Ausnahme zu genehmigen. Die Verschlüsse-

lungssoftware OpenSSL gibt das Ergebnis einer Zertifikatprüfung nur an die nutzende Anwendung weiter, fällt aber selbst keine Entscheidung über Akzeptanz oder Ablehnung. Diese Entscheidung liegt bei den nutzenden Anwendungen wie Browsern oder E-Mailprogrammen.

Beim Aufruf einer Webseite, deren Zertifikat mit einer manuell importierten CA signiert wurde, war mit den OS/Browser-Kombinationen Windows10/Internet-Explorer, Windows10/Firefox, Ubuntu/Firefox, Ubuntu/Chrome problemlose Akzeptanz (Win/IE), Ausnahmegenehmigung (Win/Firefox) und komplette Arbeitsverweigerung (Ubuntu), also auch keine Möglichkeit der Ausnahmegenehmigung, festzustellen. Die Grundphilosphie, in Problemfällen dem Nutzer die Entscheidung zu überlassen und nur beratend tätig zu werden, wird augenscheinlich inzwischen nicht mehr von allen System unterstützt. Bemerkenswert ist dabei die unterschiedliche Reaktion der neuesten Softwareversionen des gleichen Herstellers (Mozilla) und komplett unterschiedliche Fehlermeldungen von allen Systemen.

Für die Versuche war daher notwendig, Zertifikate, die den Philosophien der Software-Herstellern entsprechen, zu verwenden, um Akzeptanz zu erreichen. Das OpenSSL Handbuch der Universität Hamburg (http://www.absolute-cool.de/dokus/ssl/openssl/) listet folgende Regeln für Zertifikate auf:

> **Critical Bit**
>
> Das *Critical Bit* ist ein Flag, das für die meisten Extensions im Zertifikat gesetzt werden kann. Es wird beim Signieren durch eine CA zusammen mit der Extension gesetzt. Bei korrekter Implementierung einer Anwendung (z.B. eines Browsers) *muß diese eine als Critical markierte Extension interpretieren* können. Ist die Anwendung dazu nicht in der Lage (die Anwendung "kennt" die Extension nicht), hat sie das Zertifikat, das diese Extension enthält, zurückzuweisen. Auch dann, wenn das Zertifikat technisch korrekt ist. Das Critical Bit soll also eine Möglichkeit bieten, eine bestimmte Verwendung eines Zertifikats zu erzwingen.
>
> Da Zertifikate, die dieses Bit gesetzt haben, zurückgewiesen werden können, sollte der Einsatz des Critical Bit gut überlegt werden. Es gibt beispielsweise verschiedene Ansichten über die Verwendung der einzelnen Key Usage Attribute, so daß eine Critical-Markierung hier nicht ohne "Risiko" ist.
>
> **Basic Constraints**
>
> Mittels Basic Constraints kann eine Anwendung erkennen, ob es sich bei einem Zertifikat um ein CA-Zertifikat handelt oder nicht. Diese Extension sollte in jedem CA-Zertifikat verwendet und als Critical markiert werden, auch wenn möglicherweise einige Anwendungen wegen der Critical-Markierung das Zertifikat zurückweisen.
>
> Basic Constraints besteht aus einem Feld `cA`, welches ein BOOLEAN ist, sowie einem optionalen INTEGER-Feld, `pathLenConstraint`. Für CA-Zertifikate muß das `cA`-Feld auf TRUE gesetzt werden, für andere auf FALSE. Laut RFC 2459 sollte Basic Constraints in Nicht-CA-Zertifikaten nicht verwendet werden, also auch nicht, wenn die Extension FALSE markiert ist. `pathLenConstraint` ist nur sinnvoll in CA-Zertifikaten und gibt an, wieviele CA-Ebenen unterhalb des CA-Zertifikats maximal zulässig sind. Ein Wert 0 bedeutet hierbei, diese CA gibt nur Anwendungs- bzw. Benutzerzertifikate heraus. Basic Constraints sollte immer als Critical markiert werden.
>
> Laut Stephen N. Henson ist die Basic Constraints Extension unbedingt erforderlich in CA-Zertifikaten, die S/MIME Benutzer zertifizieren. Andernfalls wird die S/MIME-Mail beim Empfänger aufgrund eines ungültigen CA-Zertifikats zurückgewiesen.

Zum Thema „Basic Constraints“ verweist dieses Papier ebenfalls auf unterschiedliche Verhaltensweisen verschiedener Browser. Ähnliches gilt für den folgenden Abschnitt

> **Key Usage**
>
> Key Usage steuert den Verwendungszweck des zu einem Zertifikat gehörenden Schlüssels: z.B. darf ein Schlüssel nur zum Signieren von CRL's, nur zur Daten-Verschlüsselung oder nur zum Unterschreiben verwendet werden. Laut Netscape Dokumentation vom 13.08.97, Netscape Certificate Extensions - Communicator 4.0 Version, wird Key Usage vom Netscape Browser (NSC) zur Beschränkung der Verwendung von Zertifikaten ausgewertet. Allerdings nur, wenn Key Usage als **Critical** (s.u.) markiert ist. Liegen andererseits mehrere Zertifikate vor (z.B. eines zum Signieren, eines zum Verschlüsseln), bestimmt Key Usage (Critical oder nicht), welches Zertifikat vom Browser verwendet wird.
>
> Laut Microsoft-Dokumentation Structuring X.509 Certificates for Use with Microsoft Products wertet der MS Internet Explorer (MSIE) Key Usage aus, egal ob Critical oder nicht. Das deckt sich aber nicht ganz mit den gemachten Erfahrungen (siehe unten).
>
> Üblicherweise wird Key Usage eingesetzt, um die Verwendung des Public Keys (und somit auch des Private Keys), an

den diese Extension durch die Zertifizierung gebunden wird, zu beschränken. *Das funktioniert natürlich nur, wenn die Applikation, mit der das Zertifikat verwendet wird, diese Extension auch interpretiert*. Der Netscape Browser (4.06) beispielsweise verweigert den Kontakt zu einem SSL-Server, wenn im Server-Zertifikat das Flag für Digital Signature gesetzt *und* dieses Critical markiert ist. Laut Netscape Dokumentation sollte dieses Flag in einem SSL-*Client*-Zertifikat gesetzt sein. Für einen SSL-Server hätte dagegen Key Encipherment gesetzt sein müssen. Der Browser läßt daher keine SSL-Verbindung zu und bricht den Verbindungswunsch mit der Meldung "The certificate is not approved for the attempted operation" ab. Dasselbe Zertifikat wurde aber vom Microsoft-Browser (Ver. 4.01) problemlos akzeptiert. In der Microsoft-Dokumentation Structuring X.509 Certificates for Use with Microsoft Products vom 4.12.97 steht im Abschnitt zum Thema Key Usage: "The only Microsoft Application that currently enforces KeyUsage is Microsoft Outlook."

Die nachstehende Beschreibung erfolgt in Anlehnung an die oben erwähnte Netscape-Dokumentation und den RFC 2459. Es stehen neun Werte für das Schlüsselwort `keyUsage` in der Konfigurationsdatei `openssl.cnf` zur Verfügung:

Bezeichnung	Wert für `keyUsage` in `openssl.cnf`	Verwendung des Public Keys
Decipher Only	`decipherOnly`	Ist *Key Agreement* gesetzt, darf der Public-Key innerhalb eines Schlüsselaustausches zur Entschlüsselung von Daten verwendet werden. Andernfalls undefiniert.
Encipher Only	`encipherOnly`	Ist *Key Agreement* gesetzt, darf der Public-Key innerhalb eines Schlüsselaustausches zur Verschlüsselung von Daten verwendet werden. Andernfalls undefiniert.
CRL Signing	`cRLSign`	Public Key kann verwendet werden, um CRLs zu verifizieren.
Key Cert Sign	`keyCertSign`	Public Key kann verwendet werden, um Zertifikate zu verifizieren.
Key Agreement	`keyAgreement`	Zur Verwendung beim Schlüsselaustausch.
Data Encipherment	`dataEncipherment`	Zur Verschlüsselung von „normalen“ Daten, also keinen Schlüsseln.
Key Encipherment	`keyEncipherment`	Public Key wird zum Schlüsselmanagement verwendet.
Non Repudiation	`nonRepudiation`	Key zur Prüfung von „bewußten“ Signaturen (außer CRLs und bei Zertifikaten).
Digital Signature	`digitalSignature`	Key zur Prüfung von „automatisierten“ Signaturen (außer bei CRLs und bei Zertifikaten).

Extended Key Usage

Extended Key Usage kann ergänzend oder anstelle von Key Usage verwendet werden. Die Extension beschränkt analog Key Usage die Verwendung des zertifizierten Public-Keys. Beispielsweise könnte die Verwendung von CA-Zertifikaten, welche die entsprechende Basic Constraints und Key Usage Extension enthalten, feiner unterschieden werden. Extended Key Usage könnte dazu auf `serverAuth` gesetzt sein, so daß der CA-Schlüssel lediglich zum Überprüfen von SSL-Server- aber nicht von SSL-Client-Zertifikaten dient.

Auch Extended Key Usage kann `critical` gesetzt sein und es gilt das im Abschnitt Critical Bit gesagte.

Jede Organisation kann eigene Werte für die Extension Extended Key Usage registrieren lassen. Unter anderem Netscape und Microsoft haben das getan.

OpenSSL kennt die in der Tabelle aufgeführten symbolischen Werte für das Schlüsselwort `extendedKeyUsage`. Es können aber auch andere Werte für die Extension durch Angabe der entsprechenden OID gesetzt werden.

Bezeichnung	Wert für `extKeyUsage` in `openssl.cnf`	Verwendung des Public Keys
RFC2459-Extensions		
TLS Web Server Authentication	`serverAuth`	Authentisierung von Web-Servern durch Web-Clients
TLS Web Client Authentication	`clientAuth`	Authentisierung von Web-Clients durch Web-Server
Code Signing	`codeSigning`	Key zur Signierung von Programm-Code
Email Protection	`emailProtection`	Key zur Verwendung mit S/MIME-Software
Time Stamping	`timeStamping`	Signierung von Objekt-Hashwerten und zugehörigen vertrauenswürdigen Zeitstempeln
Microsoft-Extensions		
Individual Code Signing	`msCodeInd`	
Commercial Code Signing	`msCodeCom`	
Trust List Signing	`msCTLSign`	
Server Gated Crypto	`msSGC`	Server-Zertifikat mit „Global Server ID“
Encrypted File System	`msEFS`	Verschlüsselung von symmetrischen Keys zur Dateissystem-Verschlüsselung
Netscape-Extensions		
Server Gated Crypto	`nsSGC`	Server-Zertifikat mit „Global Server ID“

In der Microsoft-Dokumentation Structuring X.509 Certificates for Use with Microsoft Products vom 4.12.97 steht, daß für den Einsatz von Zertifikaten im Zusammenhang mit Microsofts "Authenticode" nur der Wert `codeSigning` für Extended Key Usage angegeben sein darf. Ebenfalls in dieser Dokumentation steht, daß die Gültigkeit von Extended Key Usage im Anwendungszertifikat nur gegeben ist, *wenn sämtliche Zertifikate der Zertifikatkette diese Extension enthalten*. Ist die Extension dagegen garnicht enthalten, sollen MS-Anwendungen das Zertifikat für jeden durch Key Usage beschriebenen Zweck als gültig interpretieren.

Auch Netscape scheint das Setzen der Extension in allen Zertifikaten der Kette zu unterstützen. Es scheint aber doch fraglich, wieviel Sinn es beispielsweise macht, in einen CA-Zertifikat Extended Key Usage auf `email` zu setzen, damit die Extension in einem Anwendungszertifikat gültig ist. Die Empfehlungen von Netscape für diese Extension können in Installation and Deployment Guide, Appendix B nachgelesen werden.

Zu Netscapes bzw. Microsofts SGC ("Server Gated Cryptography") ist anzumerken, daß die Verwendung dieser Werte nur im Zusammenhang mit speziellen CA-Zertifikaten sinnvoll ist. Diese CA-Zertifikate werden durch ein Flag *im Browser* als für SGC geeignet gekennzeichnet. Ein eigenes CA-Zertifikat kann nur nach Import in den Browser und anschließendem Patchen der Zertifikat-Datenbank (Netscape) mit diesem Flag ausgestattet werden. Genaueres steht in der Datei `README.GlobalID` des SSL-Apache-Pakets ModSSL

Eine Empfehlung für die Werte dieser Extension ist nur schwer zu geben, gerade wegen der MS-Forderung, daß die Extension in allen Zertifikaten der Kette enthalten sein muß. Am sinnvollsten scheint es zu sein, ganz auf diese Extension zu verzichten.

Bei den folgenden Datenfeldern kann sich Interpretationsspielraum beim Aufbau einer EI ergeben, da die Zusammenhänge zwischen verschiedenen Versionen einer EI anders definiert sind.

Subject Key Identifier

Die Extension Subject Key Identifier enthält den Hash-Wert des Public Keys eines Zertifikates. Dadurch kann ein zu einem Public Key gehörendes Zertifikat effizient gesucht werden. So wird die Überprüfung von Zertifikatketten und bei mehreren Anwendungszertifikaten die Auswahl des richtigen Zertifikats unterstützt. Diese Extension sollte sowohl in CA-Zertifikaten als auch in Anwendungszertifikaten enthalten sein.

Authority Key Identifier

Die Extension Authority Key Identifier besteht aus drei Feldern: `keyIdentifier`, `authorityCertIssuer` und `authorityCertSerialNumber`. Laut RFC 2459 kann ein Schlüssel mittels dieser Extension auf zwei Arten identifiziert werden: entweder durch alleiniges Setzen des `keyIdentifier`-Feldes, oder durch Setzen der anderen beiden Felder. Diese Extension unterstützt die Überprüfung von Zertifikatketten.

Microsoft empfiehlt die zweite Variante, damit eine Zertifikatkette überprüft werden kann. RFC 2459 empfiehlt die erste Variante.

In der Konfigurationsdatei:

```
authorityKeyIdentifier = <[keyid[:always]][, issuer[:always]]>
```

Ist `keyid` gesetzt, wird Subject Key Identifier des Herausgeber-Zertifikats kopiert. Kann der Wert dieser Extension nicht kopiert werden (weil Subject Key Identifier nicht gesetzt war) und ist `keyid` auf `always` gesetzt, wird mit einer Fehlermeldung abgebrochen. Ist `keyid` nicht `always` gesetzt, werden alternativ das Issuer-Feld und die Seriennummer kopiert. Ist `issuer` auf `always` gesetzt, werden immer das Issuer-Feld und die Seriennummer kopiert.

Subject Alternative Name

Die Extension Subject Alternative Name kann verwendet werden, um weitere Bezeichner für ein Subject in das Zertifikat zu bringen. Es sind E-Mail-Adressen, DNS-Namen, IP-Adressen, URIs und registrierte IDs (RIDs), auch mehrfach, möglich. Diese können auch beliebig kombiniert werden.

In der Konfigurationsdatei:

```
subjectAltName=<[email:<copy|name@mail.de>][,        URL:http://my.url.here/][,
RID:1.2.3.4][, IP:1.2.3.4]>
```

Issuer Alternative Name

Die Extension Issuer Alternative Name ist wie Subject Alternative Name aufgebaut.

Hier wird allerdings nicht `email:copy` sondern `issuer:copy` unterstützt. Dabei werden die Angaben vom Subject Alternative Name des *Herausgeber*-Zertifikats kopiert.

In der Konfigurationsdatei:

```
subjectAltName=<[issuer:copy][,    URL:http://my.url.here/][,    RID:1.2.3.4][,
IP:1.2.3.4]>
```

Das folgende Feld spielt ggf. eine Rolle im Zusammenhang mit den nachfolgend beschriebenen CA-Klassen:

Certificate Policies

Die Extension Certificate Policies verweist auf die Policy, unter der ein Zertifikat herausgegeben wurde. Laut RFC 2459 sollte die Extension lediglich aus einer OID (*Object Identifier*) bestehen. Eine Anwendung, die Zertifikate prüft, sollte eine Liste mit den OID's akzeptierter Policies enthalten und diese gegen die Policy-OID eines Zertifikats vergleichen. Dann wird das Zertifikat entsprechend akzeptiert oder zurückgewiesen.

Wenn die OID nicht ausreichend ist (weil z.B. die eingesetzten Applikation keine OID-Listen unterstützen), beschreibt RFC 2459 die Möglichkeit einen URI anzugeben, der auf die Policy der CA verweist (*Certification Practice Statement, CPS*). Für Anwendungszertifikate und CA-Zertifikate, die an andere Organisationen (also nicht die der Herausgeber-CA) herausgegeben werden, kann zusätzlich eine *User Notice* festgelegt werden.

Die User Notice besteht aus zwei optionalen Feldern: `noticeRef` und `explicitText`. Das erste Feld `noticeRef` besteht aus dem Organisations-Namen und einer Nummer. Der Nummer ist ein Text zugeordnet und soll wieder durch eine Anwendung mit einer entsprechen Liste interpretiert werden. Findet eine Anwendung diese Nummer in ihrer Liste, soll sie den zugehörigen Text anzeigen. Das zweite Feld enthält einen, maximal 200 Zeichen umfassenden, frei gestaltbaren Text und steht direkt im Zertifikat. Eine Anwendung soll diesen dann bei Verwendung des Zertifikats anzeigen.

Microsoft-Anwendungen scheinen mit der Extension Schwierigkeiten zu haben, wie Peter Gutmann in seinem X.509 Style Guide beschreibt:

> "Although various MS programs give the impression of handling certificate policies, they only have a single hardcoded policy which is the Verisign CPS. To see an example of this, create a certificate with a policy of (for example) "This policy isn't worth the paper it's not written on" and view the cert using Outlook Express. What's displayed will be the Verisign CPS."

Outlook Express 5 und der Internet Explorer 5 scheinen diese Probleme nicht mehr zu haben. Eine User Notice wird für Benutzerzertifikate korrekt angezeigt. Die URL für die CPS wird ebenfalls ausgewertet und die entsprechende Seite angezeigt.

In der Konfigurationsdatei:

```
certificatePolicies=[ia5org,]OID[, OID, ...][,@polsect]
```

Laut OpenSSL-Dokumentation von Stephen N. Henson (im Quell-Verzeichnis des OpenSSL-Pakets unter `doc/openssl.txt`) ist die `ia5org`-Option für die Verwendung mit dem Internet Explorer erforderlich, obwohl sie nicht RFC-konform ist.

Die Option `@polsect` verweist auf einen Abschnitt in der Konfigurationsdatei, der die Werte für das CPS und indirekt für das User Notice Feld enthält:

```
[ polsect ]
policyIdentifier=OID
CPS="http://www.ca.de/policy.html"
userNotice=@notice

[ notice ]
explicitText="Nur zur Verschluesselung von E-Mail (S/MIME)"
organisation="CA Org."
noticeNumbers=4, 2
```

Eine Rolle bei der Akzeptanz von Zertifikaten scheint die Klasse des Root-Zertifikats zu spielen. Die US-Regulierungsbehörden unterscheiden folgende Zertifikatklassen:

> Class 1 for individuals, intended for email.
>
> Class 2 for organizations, for which proof of identity is required.
>
> Class 3 for servers and software signing, for which independent verification and checking of identity and authority is done by the issuing certificate authority.
>
> Class 4 for online business transactions between companies.
>
> Class 5 for private organizations or governmental security

Trotz vieler Diskussionen in Internetforen wird allerdings nicht klar, wie sich Zertifikate verschiedener Klassen unterscheiden. Bei Inspektion der CA aus den Zertifikatverwaltungen zeigt sich:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 8069548958653521 (0x1cab36472d9c51)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=IL, O=StartCom Ltd., OU=Secure Digital Certificate Signing,
                CN=StartCom Certification Authority
        Validity
            Not Before: Oct 14 20:57:09 2007 GMT
            Not After : Oct 14 20:57:09 2022 GMT
        Subject: C=IL, O=StartCom Ltd., OU=Secure Digital Certificate Signing,
                 CN=StartCom Class 2 Primary Intermediate Server CA

-----------------------------------------------------------------------------

Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number:    70:ba:e4:1d:10:d9:29:34:b6:38:ca:7b:03:cc:ba:bf
        Signature Algorithm: md2WithRSAEncryption
        Issuer: C=US, O=VeriSign, Inc.,
                OU=Class 3 Public Primary Certification Authority
        Validity
            Not Before: Jan 29 00:00:00 1996 GMT
            Not After : Aug  1 23:59:59 2028 GMT
        Subject: C=US, O=VeriSign, Inc.,
                 OU=Class 3 Public Primary Certification Authority
```

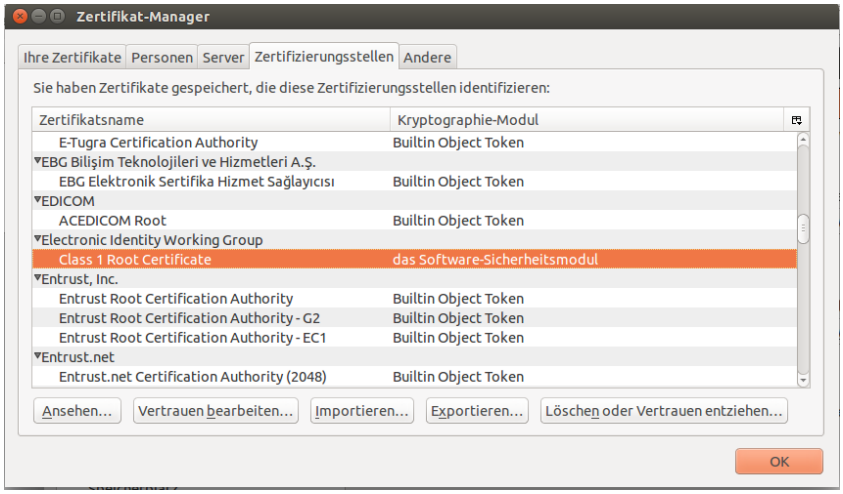
Mehr als dass „Class x“ in verschiedenen Datenfeldern auftaucht, ist an Systematik nicht zu finden. Aufgrund der Anmerkungen über „CertificatePolycies“ im OpenSSL-Handbuch könnte man vermuten, dass dieses Feld eine Rolle bei der Festlegung der Klassen spielt, jedoch finden sich Angaben zu Polycies nur bei wenigen CAs. Anfragen bei verschiedenen Herausgebern offizieller CA-Zertifikate führten nicht zu einer Klärung, sondern – wie leider heute in vielen Bereichen üblich – zu einer Wiederholung der Frage ohne Fragezeichen, d.h. zu der Auflistung der Klassen wie in der obigen Abbildung, bzw. – auch aufschlussreich über die Zustände in der Branche – in zwei Fällen zu HTTP-Error 404 beim Versuch, das Webmail-Formular der CA auf deren Webseite zu erreichen. Irgendein bindendes Sicherheitskonzept ist mit dem Klassenkonzept, so wie es sich präsentiert, jedenfalls nicht verbunden: jeder

kann beliebige Einträge in ein eigenes CA-Zertifikat übernehmen, und so lange keine übergeordnete Agentur wiederum die CA signiert, ist keine Kontrolle möglich.

Steckt Verborgenes dahinter? Wenn man sich die Menüführung in den Zertifikatverwaltungen anschaut, stellt man fest: im Prinzip ist die verkaufte Sicherheit nicht erzielbar, wenn die Software wirklich das macht, was die Bedienoptionen anbieten. Dem Nutzer stehen eine Reihe von Bedienoptionen zur Verfügung, die Kenntnisse erfordern, die über die eines normalen Nutzers hinausgehen, aber in keiner Weise so abgesichert sind, dass auch unerfahrene Nutzer die notwendige Kontrolle behalten. Andererseits ist das X.509-Zertifikat ein gutes Geschäft für die CA, das man sich nicht verderben will. Es sieht danach aus (Reaktionen der Anwendungen, verschiedene Softwaremodule usw.), dass die Software-Entwickler eine ganze Menge in der Hintergrundsoftware abwickeln, was nicht zu den Funktionen passt, die dem Nutzer angeboten werden, die man aber auch nicht ersatzlos durch das Sledge Hammer-Motto „Vertrauen Sie mir! Ich weiß, was ich tue!“ ersetzen will, denn der völlig entmündigte Nutzer ist sicher auch kein guter Geschäftspartner. Als Konsequenz scheinen sich Softwareablauf und Nutzermenü in manchen Fällen gegenseitig zu torpedieren, und böse Stimmen in Foren behaupten denn auch, die Konstruktion der Browser orientiere sich eher an wirtschaftlichen statt an technischen Regeln.

Die Zertifikatverwaltung

Das desolate Bild, das sich bereits nach Inspektion der allgemeinen Zertifikatpraxis bietet, setzt sich fort, wenn man in Details einsteigt. Eine Zertifikatverwaltung kann von verschiedenen Anwendungen aus angesprochen werden: in Linux beispielsweise, mit dem wir uns exemplarisch beschäftigen wollen, Thunderbird, Firefox, Chrome, Seahorse, Kleopatra, … Die Zertifikatverwaltung erscheint hierdurch zentralisiert und nach Nutzerpräferenz gut nutzbar zu sein. Bei Versuchen stellte sich allerdings heraus, dass Zertifikate, die mit einer Anwendung importiert wurden, nur teilweise in den anderen Anwendungen ebenfalls zur Verfügung standen, teilweise aber auch nicht und nochmals importiert werden mussten. Regeln für dieses Verhalten konnten anhand des Zertifikataufbaus nicht abgeleitet werden. Verschiedene Anwendungen scheinen in unterschiedlichen Verzeichnissen nach Zertifikaten zu suchen: neben dem physischen Speicherort (z.B. /usr/share/ca-certificates/mozilla) existieren Verzeichnisse mit Softlinks in die Zertifikatverzeichnisse (z.B. /etc/ssl/certs), was Möglichkeiten zu Abweichungen eröffnet, und möglicherweise verfügen selbst auf die gleichen Verzeichnisse zugreifenden Anwendungen über eigene nachgeschaltete Datenbanken. Beispielsweise war das in Thunderbird importierte in der Abbildung markierte Root-Zertifikat dort zu finden, in der Firefox-Verwaltung – Produkt des gleichen Herstellers und angeblich aus der gleichen Softwarefamilie – nicht.



Der Screenshot zeigt eine weitere Merkwürdigkeit: die Zertifikate werden von unterschiedlichen Softwaremodulen verwaltet. Solche Redundanzen sind ein absolutes MUST NOT in der Softwaretechnologie, und es erstaunt schon etwas, dass das offiziell zugegeben wird.

Die Anwendungen erlauben beim Import eines Zertifikats die Festlegung eines Vertrauensniveaus, halten sich jedoch im Betrieb nicht an die Vorgaben des Nutzers. Von diesem im „Software-Sicherheitsmodul“ als „vertrauenswürdig“ gekennzeichnete Zertifikate werden trotzdem als „unbekannt“ oder „unsicher“ bezeichnet.

Konkrete Versuche mit Zertifikaten

Bei Versuchen zeigt sich ziemlich schnell (auch wieder am Thunderbird/Ubuntu-Beispiel), dass mit selbst signierten Zertifikaten, wie vom EI-Prinzip vorgesehen, gar nichts zu erreichen ist. Die X.509-Prüfungsmodalitäten bei E-Mails sind schwächer als bei HTTPS, da nur ein Class 1-Root-Zertifikat verwendet werden muss (der Zertifizierer prüft nicht die Identität des Zertifikatinhabers, sondern bindet das Zertifikat nur an eine E-Mail-Adresse), während bei HTTPS ein Class 3-Zertifikat angestrebt wird (Überprüfung der Identität, aber ebenfalls nicht bindend). Die X.509-Absicherung über eine CA ist damit nicht stärker als ein selbst signiertes Zertifikat, da auch dieses die E-Mail-Adresse enthalten muss, d.h. von der Sicherheitsphilosophie stünde einer Verwendung solcher Zertifikate nichts im Weg.

Trotzdem werden eigene Zertifikate für die Signatur von E-Mails zwar meist installiert, fremde selbstsignierte Zertifikate, die für die Verschlüsselung notwendig sind, werden jedoch zurückgewiesen, d.h. E-Mails können zwar signiert, aber nicht verschlüsselt werden. Die Reaktionen der Zertifikatverwaltung sind noch nicht einmal eindeutig:

- „Dem Zertifikat (in einer empfangenen Mail) wird nicht vertraut“, und damit verbunden ist, dass man es sich noch nicht einmal ansehen kann, d.h. die Mail ist signiert, die Signatur ist jedoch nicht prüfbar. Die gleiche Anwendung (Thunderbird) hat das Zertifikat jedoch zuvor mitsamt dem privaten Schlüssel für Signaturzwecke akzeptiert und sogar eine Signatur erzeugt.
- „Das Zertifikat ist (beim Import) nicht vertrauenswürdig und wird nicht installiert“. Mit privatem Schlüssel schluckt die Anwendung (Thunderbird) das Zertifikat aber problemlos.
- Man durchläuft alle Importschritte, aber es passiert nichts, weder eine Meldung noch eine Anzeige des importieren Zertifikats.

Da mit selbst signierten EI nicht weiter zu kommen ist, wurde mit einem zusätzlichen Root-Zertifikat gearbeitet. Es ist anzumerken, dass die Verwendung eines einheitlichen Root-Zertifikats für alle Ei nicht dem entwickelten EI-

Schema widerspricht, da eine Kommunikation mit allen EI, die vom gleichen Fremdzertifikat signiert sind, zulässig ist. Ein einheitliches Root-Zertifikat führt somit eine global gültige Lokalität ein. Für die EI werden folgende Nutzungen der Felder vorgesehen bzw. im Vorgriff auf EI-Erweiterungen definiert:

Bereich	**Feldname**	**Kommentar**
Inhaberdaten	commonName	Web-Server-URL, entspricht dem X.509-Standard, jedoch können mehrere commonName-Felder definiert werden, um alle URLs eines Users erfassen zu können.
	emailAddress	Wie vor, jedoch können mehrere emailAddress-Felder definiert werden.
Extensions	alternateSubjectName	Weitere URL, E-Mail-Adressen
	OID 2.16.276.99: electronicIdentity: electronic_identity_tree	Basis-OID für EI-Erweiterungen
2.16.276.99	OID 2.16.276.99.1: eiIdentifier: eiId	Eindeutiger Zufallsstring als Identifizierer für diese EI (bei Änderungen in anderen Bereichen)
	OID 2.16.276.99.2: eiSerialnumber: eiSerial	Seriennummer der EI-Version (bei Änderungen in anderen Bereichen)
	OID 2.16.276.99.10: eiDescription: eiText	Beliebiger, nutzerspezifischer Text

Die OID für die EI-Erweiterungen sind einem freien Bereich des ISO-OID-Baums entnommen. Über die definierten Erweiterungs-OIDs wird das Zertifikat als Root-Zertifikat für die elektronische Identität gekennzeichnet.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=DE, ST=NDS, L=D-26721 Emden, O=Electronic Identity Working Group,
                OU=Class 1 Root Certifier,
                CN=Class 1 Root Certificate/emailAddress=no@noname.com
                /2.16.276.99.1=DE13HDBC89829XCCXC/2.16.276.99.2=1
                /2.16.276.99.10=Open Root Certificate for central signed Electronic Identity
        Validity
            Not Before: Jun 16 09:11:00 2016 GMT
            Not After : Jun 16 09:11:00 2116 GMT
        Subject: C=DE, ST=NDS, L=D-26721 Emden, O=Electronic Identity Working Group,
                 OU=Class 1 Root Certifier, CN=Class 1 Root Certificate
                 /emailAddress=no@noname.com/2.16.276.99.1=DE13HDBC89829XCCXC
                 /2.16.276.99.2=1
                 /2.16.276.99.10=Open Root Certificate for central signed Electronic Identity
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:db:de:f9:16:1f:6e:61:bf:d0:71:d7:b1:c7:9b:
                    37:a1:f7:9f:67:c9:e4:38:7d:8c:ef:cc:44:9b:d0:
                    bc:ba:5d:dc:ad:2c:6e:14:4f:86:4c:b4:e7:c2:60:
                    e9:83:f6:b8:49:c8:2e:bd:1b:bc:43:e5:61:b1:05:
                    87:4a:4e:b1:05:b1:11:a8:8b:54:07:e4:ee:b6:b3:
                    cc:e2:95:2b:67:1e:9a:97:66:88:36:51:fb:1a:fe:
                    23:28:63:ed:bd:47:9a:9a:3b:5b:5a:56:59:39:8c:
                    96:e2:fd:73:c4:c8:1f:13:17:bd:30:a3:fc:d1:23:
                    a5:7c:4c:30:97:e5:3a:42:fb:f3:46:ed:6f:49:df:
                    07:7a:8b:a5:db:0b:aa:3a:fd:aa:1f:de:82:ba:bf:
                    5a:9e:36:49:66:3a:ee:53:6c:81:9d:93:4c:62:55:
                    f7:bf:cf:c5:11:09:0b:da:a9:a4:76:2e:93:1a:72:
                    e7:9f:09:a0:8b:b1:14:69:5d:eb:3e:23:d5:08:61:
                    b5:67:e5:a8:ae:ff:1f:15:65:6f:71:aa:4e:35:79:
                    04:1b:38:d0:e3:68:a7:c3:42:30:30:98:cf:8e:83:
```

```
                    a3:0d:42:a6:1f:0f:ea:0e:da:6e:f2:4e:b7:76:7b:
                    54:16:be:34:91:1f:e9:7f:ac:eb:2a:44:d6:af:84:
                    1d:5b
                Exponent: 65537 (0x10001)
            X509v3 extensions:
                X509v3 Basic Constraints: critical
                    CA:TRUE
                X509v3 Subject Key Identifier:
                    67:F2:A4:E8:6F:80:7C:2B:30:55:92:6E:23:A1:BB:76:F6:64:A9:7E
                X509v3 Key Usage:
                    Certificate Sign, CRL Sign
                X509v3 CRL Distribution Points:

                    Full Name:
                      URI:no_crl.com

                Netscape Cert Type:
                    SSL CA, S/MIME CA, Object Signing CA
                Netscape Comment:
                    xca ei certificate
        Signature Algorithm: sha256WithRSAEncryption
             52:b9:8c:09:62:2a:81:b7:1e:30:67:b0:ef:67:0a:2a:f8:05:
             e4:13:61:da:3b:0d:e1:39:1c:e6:67:91:db:96:3e:82:cf:fc:
             9f:6a:4c:c4:cd:c5:f4:0f:72:00:25:b6:61:c6:38:3a:29:8c:
             04:ec:c1:e8:61:61:8c:15:c1:40:ce:bc:3e:96:cd:35:1b:20:
             ff:d7:9c:34:cb:6e:ab:2d:7d:4e:3c:a3:dd:ae:0b:ca:35:d9:
             cb:3a:59:ed:2a:86:a0:b3:0c:b9:65:3e:83:e5:63:c1:cb:2c:
             f0:44:08:b7:da:ed:88:75:88:bd:91:f3:10:3d:9f:1e:1a:8c:
             6f:8f:d1:01:d8:ba:67:14:4e:4b:92:43:0b:e3:60:f2:00:b0:
             6d:a4:0a:a7:5e:b2:4d:ba:55:52:61:45:23:8c:84:b6:89:46:
             cb:8a:53:b2:55:fb:b9:80:ed:2d:af:60:63:a8:2a:c3:f5:c5:
             ac:d6:ac:90:40:cd:5a:07:54:a0:10:5b:d9:c9:60:07:df:fe:
             c0:15:f4:bc:fe:a7:19:7e:53:8a:8e:75:d4:ce:e8:4e:74:b4:
             42:65:65:95:b6:47:94:40:99:87:2a:0b:7e:f6:21:67:6d:b9:
             42:c7:87:b5:7a:91:6b:b2:86:eb:d4:c3:5a:67:d6:23:a7:09:
             ab:2f:b5:e5
```

Dieses Root-Zertifikat lässt sich problemlos in die Zertifikatverwaltungen der E-Mail-Agenten importieren, wobei das Vertrauensniveau auf „diesem Zertifikat unbedingt vertrauen“ festgelegt wird. Für die User-EI wurden weitere Zertifikate erzeugt, die mit diesem signiert wurden. Die sich hierbei stellenden Problem bestehen in der Frage

- welche Schlüsselverwendungen für eine Akzeptanz zulässig sind;
- ob und wie mehrere E-Mail-Konten einbindbar sind;
- wie zu gewährleisten ist, dass alle Nutzer akzeptiert werden.

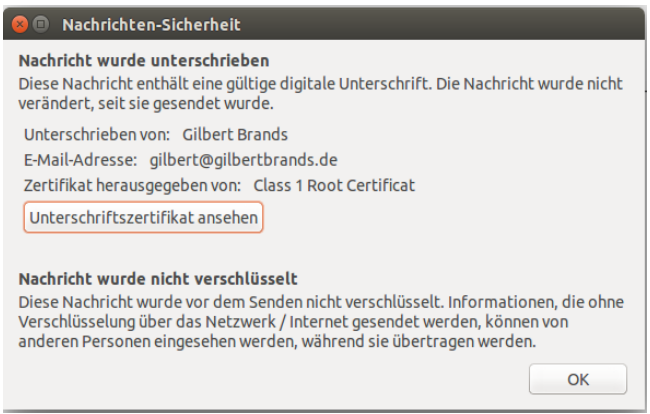
Ein zunächst funktionierendes Standardzertifikat hat folgendes Aussehen:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 02D12F7AABF2E3A2
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=DE, ST=NDS, L=D-26721 Emden, O=Electronic Identity Working Group,
                OU=Class 1 Root Certifier, CN=Class 1 Root Certificate
                /emailAddress=no@noname.com/2.16.276.99.1=DE13HDBC89829XCCXC
                /2.16.276.99.2=1
                /2.16.276.99.10=Open Root Certificate for central signed Electronic
                      Identity
        Validity
            Not Before: Jun 16 10:08:00 2016 GMT
            Not After : Dec 31 23:59:59 2035 GMT
        Subject: C=DE, L=27777 Musterstadt, OU=Musteruser, CN=www.musteruser.de
                 /emailAddress=musteruser@t-online.de/2.16.276.99.1=2389ud2jd9828989d298
                 /2.16.276.99.2=1
                 /2.16.276.99.10=Dieses Zertifikat wird von E-Mail-Agenten in der Regel
                           akzeptiert
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:da:c5:bc:c4:9a:8c:81:2b:35:64:b9:e3:71:ee:
                    c8:20:f2:24:07:60:23:1b:b8:17:aa:60:d8:cd:5a:
                    86:f9:af:48:c7:3d:17:b0:67:09:52:39:47:34:59:
                    c9:9e:f2:e5:8a:0f:d3:ef:ea:0b:51:be:73:2e:2d:
```

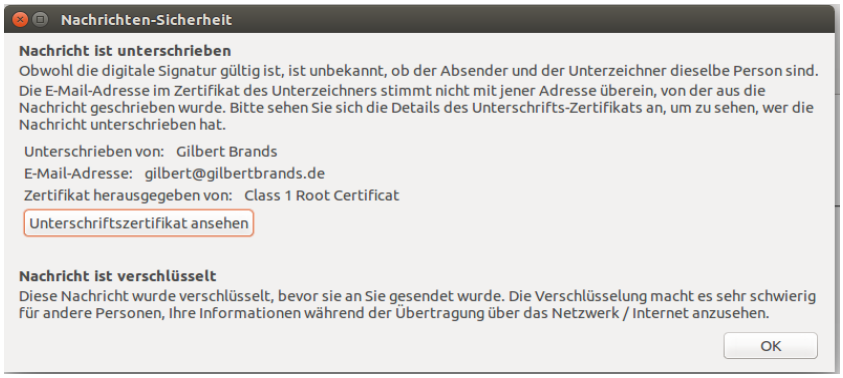
```
                01:d4:cf:8c:47:fa:3f:c1:ab:c7:91:2c:f2:6e:d2:
                a6:c9:f1:fa:31:14:6a:f6:aa:a7:b3:14:68:83:91:
                b1:94:cb:a6:d4:56:db:f0:2a:47:cc:5f:81:d1:43:
                99:ea:9e:03:12:55:38:6a:12:44:c6:88:9a:7f:45:
                5a:1c:19:65:7e:3e:06:36:2c:4e:fd:71:44:be:2d:
                b4:b0:e6:73:01:79:d6:ca:58:7e:f1:d2:d1:8a:13:
                d4:54:fb:c9:23:c3:65:0b:22:67:cb:c9:fa:e7:58:
                4c:73:9d:c6:ec:a4:38:bc:de:97:84:24:35:31:55:
                72:a9:83:c2:1c:12:5c:bb:38:f7:58:0b:eb:5d:f8:
                29:8a:72:0e:13:28:68:6f:42:d6:8d:6f:af:8a:47:
                34:7f:ee:98:fd:3b:75:b9:1e:2a:d0:0e:13:cc:2e:
                de:9d:24:0d:5f:70:10:2d:c1:f4:c6:a5:85:b4:3f:
                db:06:e1:82:b9:8b:46:17:9f:a3:54:1e:38:32:23:
                c4:0b
            Exponent: 65537 (0x10001)
    X509v3 extensions:
        X509v3 Basic Constraints:
            CA:FALSE
        X509v3 Subject Key Identifier:
            2B:9F:FD:D6:9F:75:12:A6:0F:67:01:3C:D4:DD:A2:55:CC:5D:EA:22
        X509v3 Key Usage:
            Digital Signature, Key Encipherment, Data Encipherment,
            Key Agreement
        X509v3 Extended Key Usage:
            E-mail Protection
        X509v3 Subject Alternative Name:
            email:info@secondserver.com
        Netscape Cert Type:
            S/MIME
        Netscape Comment:
            xca root signed user ei certificate
Signature Algorithm: sha256WithRSAEncryption
     bd:39:eb:a9:e4:6f:86:33:89:94:e7:cd:ec:d8:3e:40:d3:2b:
     8e:be:c0:18:81:68:fa:f9:f1:02:ce:b8:4a:82:b6:d0:b1:7f:
     9d:f0:8c:c3:c5:6a:76:55:0b:70:39:58:a1:57:a2:dc:77:da:
     16:05:a5:83:f1:c1:51:4b:53:7a:1c:85:70:98:b0:ff:a3:8a:
     de:fc:df:e5:ec:e8:42:dd:28:12:11:66:4a:1e:2f:68:33:63:
     52:11:57:2e:4d:1a:ea:25:b6:36:c5:f8:d8:17:96:86:01:9d:
     52:cb:59:eb:d7:a8:d0:3e:7f:71:8d:69:8a:c1:8c:38:9b:ca:
     c8:4b:a0:e5:d0:fa:86:ee:72:78:5e:37:35:01:f9:94:c3:4f:
     ed:e2:c6:68:7f:5f:64:5d:07:ca:d4:c6:19:33:b5:eb:a8:0e:
     f3:8a:90:b4:50:bb:c8:d8:97:0d:f4:18:66:55:c9:f1:c8:12:
     70:6e:12:99:b5:fa:f2:99:a7:6b:31:f1:e7:da:c2:16:42:be:
     18:5e:7c:21:bf:46:d0:5b:de:49:4f:99:0b:f1:a7:db:38:be:
     6b:57:d4:36:40:ad:19:30:4f:ca:96:1f:4c:73:79:ce:2f:76:
     91:dc:87:c0:d1:43:e3:d4:90:43:7d:34:30:d8:cf:89:56:2a:
     cf:6c:3e:37
```

Solche Zertifikate lassen sich sowohl als eigene Zertifikate (mit privatem Schlüssel) als auch als Personenzertifikate (an andere Teilnehmer verteilte Zertifikate zum Erzeugen verschlüsselter E-Mail) installieren. Zusätzlich ist natürlich das Root-CA-Zertifikat zu installieren. Die Idee dahinter ist, dass Root-Zertifikat mitsamt dem privaten Schlüssel in einer XCA-Datenbank allen Interessenten zur Verfügung zu stellen, so dass jeder in der Lage ist, unabhängig von Dritten gültige Zertifikate zu erzeugen und zu nutzen. Die Ausschaltung der offiziellen CA führt zu einfacherem, auch für private Nutzer durchschaubarem Management, höherer Sicherheit, Beteiligung und damit mehr Verantwortung für die Nutzer. Nichtteilnehmer wären von dem Verfahren nicht betroffen, da das Root-Zertifikat nur bei Teilnehmern installiert ist und der Erhalt verschlüsselter oder signierter Mail ohne Root-Zertifikat zu den üblichen Warnmeldungen der Systeme führen würde.

Was bei der Konfiguration allerdings noch vernünftig aussieht, entwickelt sich in der Anwendung erneut zu einem Disaster. Wird eine E-Mail signiert (oder signiert und verschlüsselt), sehen Sender und Empänger verschiedene Informationen. Beim Sender scheint alles in Ordnung:



Beim Empfänger hingegen werden Warnungen ausgegeben, obwohl das Root-Zertifikat installiert und als vertrauenswürdig gekennzeichnet ist. An dieser Warnung ändert sich auch dann nichts, wenn das Zertifikat manuell als „vertrauenswürdig“ gekennzeichnet wird. Die Anwendung weigert sich schlicht, das zur Kenntnis zu nehmen.



Noch merkwürdiger wird die Reaktion des Systems, wenn das Fremdzertifikat nicht manuell installiert, sondern wie üblich mit einer Mail verschickt wird, um es dem Empfänger mitzuteilen. Thunderbird behauptet, das Root-Zertifikat nicht zu kennen, obwohl es installiert ist, lehnt selbst eine Darstellung des Zertifikatinhaltes ab und löscht sogar ein bereits manuell erfolgreich installiertes Fremdzertifikat ohne jegliche Rückmeldung aus der Verwaltung.

Die Versuche wurden mit verschiedenen Zertifikateinstellungen wiederholt, wobei funktionierende Zertifikate als Grundlage genommen wurden. Dabei konnte beobachtet werden, dass selbst Zertifikate von CAcert, einer größeren Organisation, die sich die Verbreitung kostenloser Zertifikate zum Ziel gesetzt hat, deren Root-Zertifikate aber nie offiziell akzeptiert wurden und daher manuell installiert werden müssen, nicht einwandfrei funktionieren. Auf Zertifkate von Let's Encrypt, das ähnliche Konzepte, aber mit wesentlich weniger eingebauter Sicherheit bezüglich der Identitäten verfolgt, scheint das nicht zuzutreffen. Deren Root-Zertifikate sind im Gegensatz zu CAcert anerkannt, allerdings steckt dort das US-Verteidigungsministerium DoD dahinter, was bedenklich stimmen sollte.

Als Schlussfolgerung bleibt selbst bei „den guten Jungs“ von Mozilla nur:

- Die Software macht nicht das, was die Bedienoberfläche an Funktionen anbietet und die X.509-Grundphilosophie vorgibt.

Es mag ein wenig hart klingen, aber aus Sicht ordentlicher Software-Technologie wird die Nutzung des X.509-Systems außerhalb der Kontrolle offizieller CA eindeutig sabotiert.

Andere Systeme

Das endgültige Aus für diesen Weg, allgemeine Verschlüsselung auf den Weg zu bringen, kommt bei Betrachtung weiterer Systeme wie Windows/Mail oder Android und Tablets/Phones. S/MIME, die E-Mail-Version des X.509-Systems, war erfahrungsgemäß und im Gegensatz zu PGP über mehr als ein Jahrzehnt hinweg Standardbestandteil der E-Mail-Anwendungen. Man musste nur Zertifikate installieren und konnte es nutzen. Eine Inspektion ergab jedoch:

- Unter Windows10 steht S/MIME nicht mehr zur Verfügung bzw. nur noch dann, wenn das Standard-Mail-Programm mit dem IMAP-Protokoll verwendet wird (die Installation anderer Programmtypen haben wir nicht berücksichtigt, weil der normale Internetnutzer dies auch nicht tun würde). Beim IMAP-Protokoll werden die Mails allerdings zentral auf einem Server verwaltet, für private Nutzer in der Regel einem MicroSoft-Server (Unternehmen können sich ntürlich eigene Hardware und die Software kaufen).

Im Klartext: unter Windows ist eine Verschlüsselung von E-Mails nur dann möglich, wenn man sie im Klartext auf einem MicroSoft-Server speichert. Mindestens der Provider kann die E-Mails lesen. Verschlüsselung wird damit so sinnvoll wie der Versuch, einen bei Hochwasser vollgelaufenen Keller mit dem Suppenlöffel leer zu schaufeln.

- Mail-Anwendungen in Android sehen S/MIME (oder PGP) gar nicht erst vor. Zusätzliche Apps können das zwar beseitigen, bringen aber neue Probleme und Inkompatibilitäten mit anderen Anwendungen mit sich.

Selbst wenn Zertifikate im angestrebten Sinn nutzbar wären, zeigt sich bei diesen Systemen, dass die Bedienbarkeit auf jeden Fall weit jenseits dessen liegt, was man einem normalen Nutzer zumuten kann. Selbst für Experten und Nerds ist die Bedienung teilweise völlig undurchschaubar. Um zu den erweiterten Einstellungen bei der Einrichtung des Windows-Mailers zu gelangen, die notwendig sind, wenn man Mailkonten bei anderen Providern nutzt, ist beispielsweise dreimal kurz hintereinander „Verbinden“ zu drücken. Weiß man das nicht, erhält man nur die Rückmeldung „Server inkompatibel, fragen Sie Ihren Provider“ (erfahrungsgemäß trifft man dort aber nicht auf Leute, die solche Probleme lösen könnten).

Fazit

Der zunächst sinnvolle Versuch, vorhandene Technologie zu nutzen, ohne deren Einsatzphilosophie zu verletzen, scheitert daran,

- dass sich die Softwarehersteller nicht an die eigene Philosophie halten,
- dass die Bedienung derart gestaltet ist, dass eine nutzerfreundliche Bedienung ausgeschlossen ist,
- dass vorhandene Technologie aus den Softwarepaketen entfernt wird.

Obwohl klar ist, dass insbesondere E-Mails dringend der Verschlüsselung bedürfen, wird anscheinend ziemlich viel getan, das zu verhindern. Verschlüsselung ja, aber nur wenn man zahlt. Ansonsten ist Verschlüsselung trotz aller Beteuerungen unerwünscht.

Der folgende Abschnitt soll zeigen, wie einfach ein Einstieg sein könnte, wenn die Software tatsächlich das machen würde, was sie soll. Es handelt sich um eine Bedienungsanleitung für einen interessierten, aber technisch ungeschulten Nutzer.

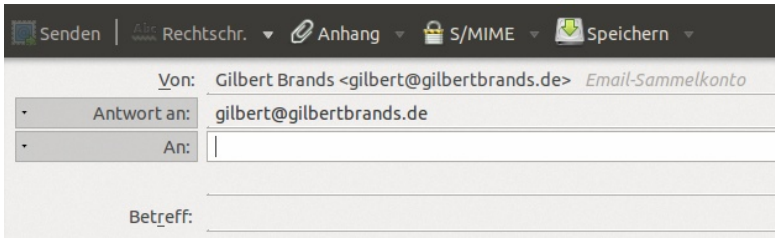
α – Kevin und die elektronische Identität, Teil 1

Hallo

Du bist es leid, dass jeder deine E-Mails mitlesen kann ? Dass du nur beim BND anrufen musst, um den Inhalt einer zerstörten E-Mail wieder herzustellen ? Dann solltest du deine E-Mails verschlüsseln. Und zwar nicht nur zwischen dir und dem Server deines Providers, sondern zwischen dir und dem Empfänger. Wir zeigen dir hier in einigen wenigen Schritten, wie das geht.

Elektronische Identität

Für die Verschlüsselung von E-Mails gibt es zwei Konzepte: S/MIME mit X.509-Zertifikaten und PGP. S/MIME kennst du vielleicht vom Fenster deines E-Mailprogramms:



Beide sind kompliziert, und S/MIME kostet meist auch Geld. Wir haben das Konzept der „elektronischen Identität“ (EI) entwickelt, dass auf S/MIME aufsetzt. Was wir dir hier zeigen, ist noch nicht die EI, dazu gehört noch ein wenig mehr. Das hier ist sehr einfach und verschlüsselt deine Mails sicher, und wenn du Spaß dran hast, kannst du das EI-Konzept ja weiter verfolgen und ausbauen.

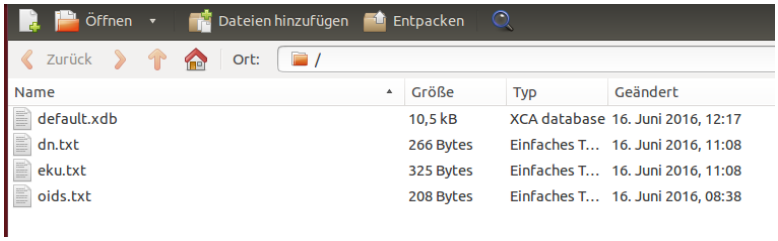
XCA

Zunächst brauchst du ein Programm names XCA. Keine Bange, das kostet nichts, weil es OpenSource-Software ist. Installiere es:

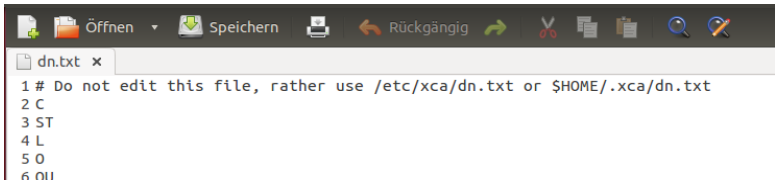
- Linux: XCA ist in der Paketverwaltung vorhanden, du brauchst es nur in der Paketverwaltung zu suchen und zu installieren.
- Windows: fertige Binaries sind aus Internet downloadbar.

Die Basisdateien

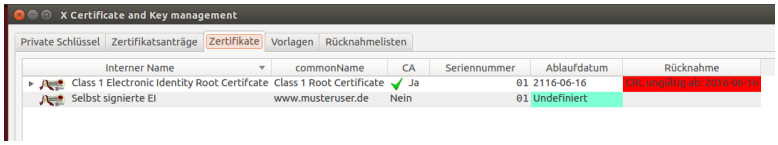
Installiere die Dateien unseres Pakets. Wenn du die ZIP-Datei öffnest, siehst du die Liste. Es sind nur 4 Stück:



Wo? Suche auf deinem System beispielsweise nach der Datei „dn.txt“ und öffne sie mit einem Texteditor. Dort findest du (das hier ist Linux)



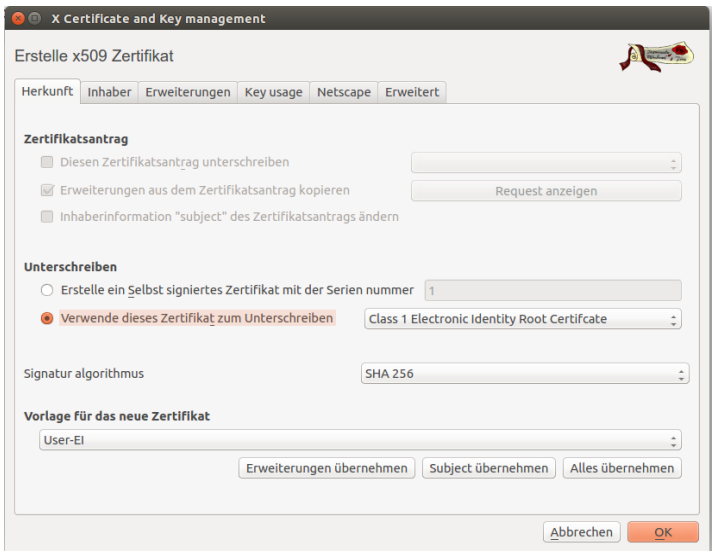
Lege ggf. ein Verzeichnis „/etc/xca“ an und kopiere sie hinein. Öffne dann XCA, wähle „Datenbank öffnen“ und gebe „default.xdb“ aus dem Verzeichnis, in das du die Dateien kopiert hat ein. Siehst du dann



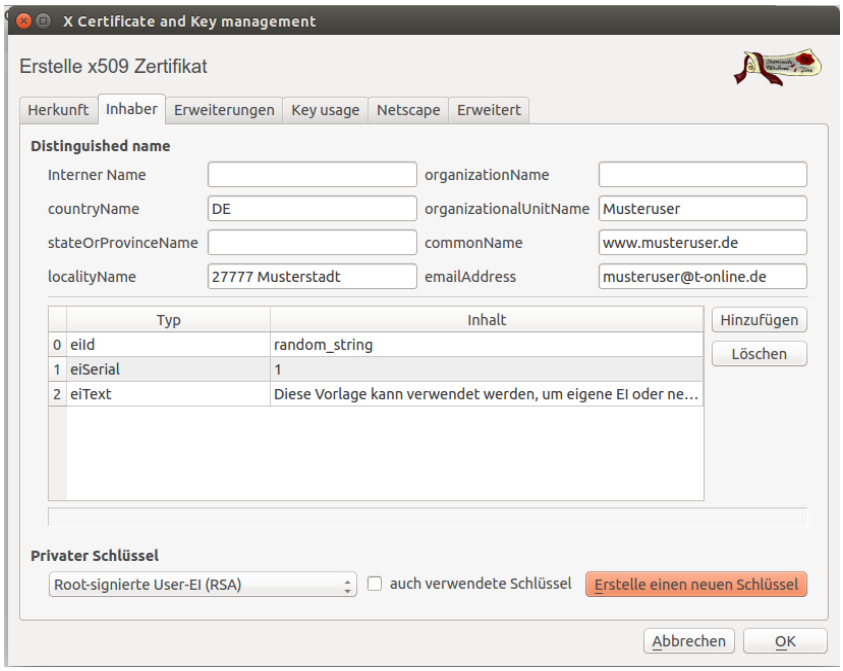
hat alles funktioniert und wir können mit den Verschlüsselungsteilen beginnen.

Ein EI-Zertifikat erzeugen

Wähle im Menü „Zertifikate“ den Punkt „neues Zertifikat“ aus. Es öffnet sich ein Fenster mit dem Inhalt

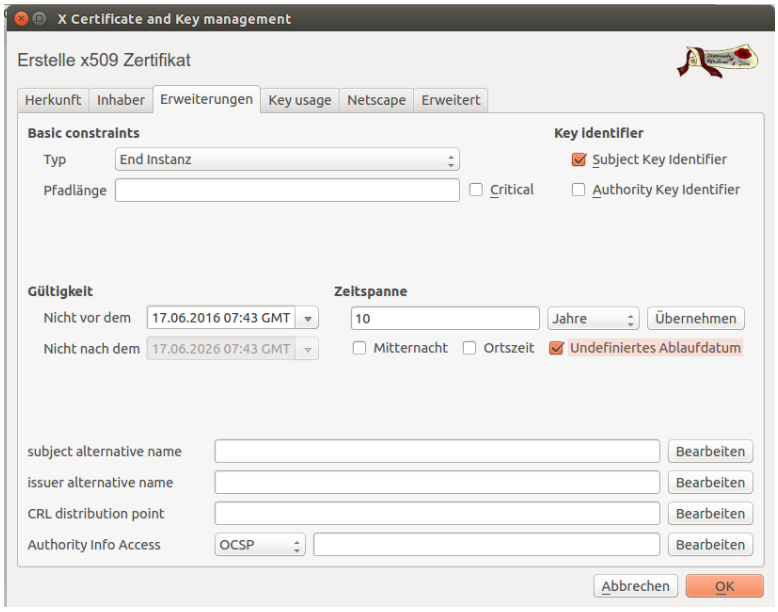


Klicke alles so an, dass das Bild dem Screenshot entspricht und klicke abschließend „alles übernehmen“, Dann kannst du zum Menü „Inhaber“ weitergehen. Hier findest du, wenn alles korrekt gelaufen ist

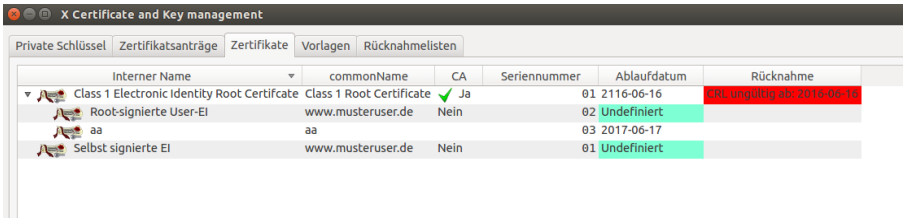


Gebe einen „internen Namen“ an, unter dem du später dein Zertifikat findest. Beispielsweise „E-Mail-EI für mich“ oder irgendetwas anderes. Auch die restlichen Felder kannst du so füllen, dass du daran erkennbar bist. Bei „commonName“ gibt deinen Namen oder deinen Spitznamen an. Wichtig:

- Gib bei „emailAddess“ die E-Mail-Adresse an, über die die verschlüsselte Mail laufen soll. Im Moment funktioniert das nur für eine und für genau diese Adresse.
- Klicke auf „Erstelle einen neuen Schlüssel“ und gebe eine Länge von 2.048 Bit an. Wie du ihn nennst, kannst du wieder selbst bestimmen.

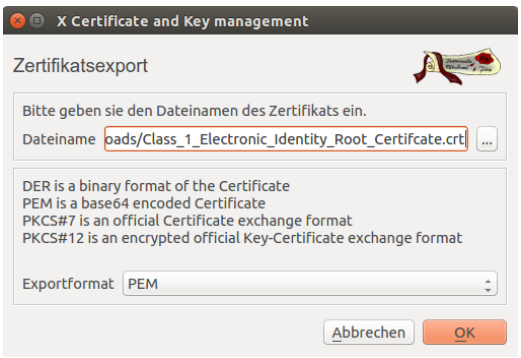


Den Rest kannst du ändern, musst es aber nicht. Das war es schon. Bei den anderen Menüpunkten musst du nichts machen, aber schau es dir ruhig alles einmal an. Drück nun einfach ok, und im Hauptmenue sollte nun dein neues Zertifikat auftauchen:

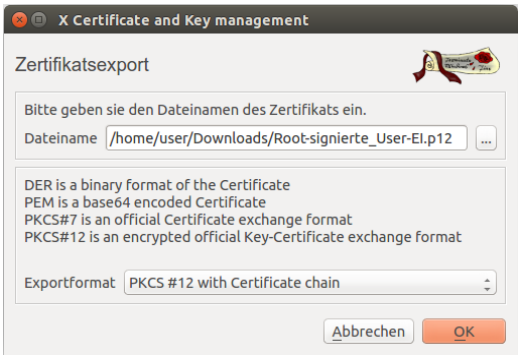


Thunderbird – E-Mail-Programm: Installation der Zertifikate

Markiere zunächst das Zertifikat „Class 1 ...“ und klicke auf Export.



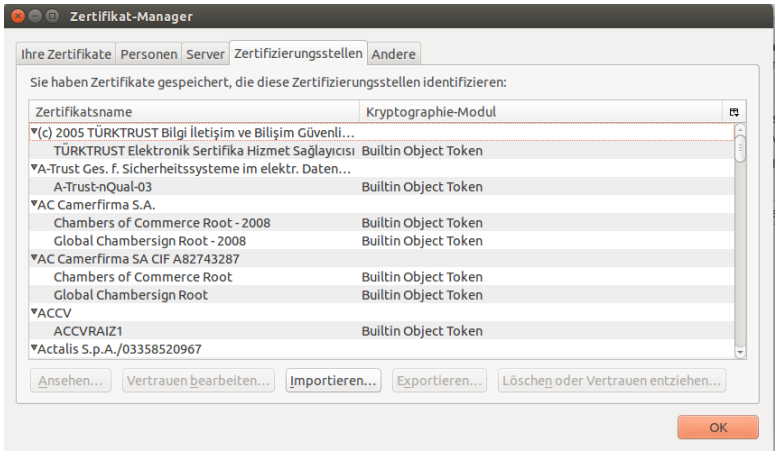
Merke dir, wohin es gespeichert wird. Den Namen und das Verzeichnis kannst du auch aussuchen. Markiere dann dein Zertifikat und exportiere es ebenfalls, nun aber als .p12-Zertifikat



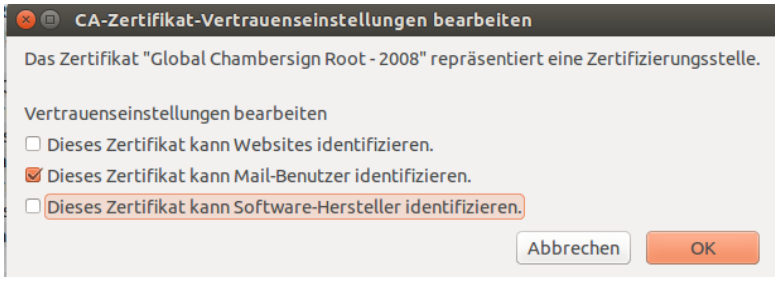
Dabei wirst du nach einem Kennwort gefragt. Du kannst einfach „1“ oder „a“ nehmen. Danach kannst du XCA schließen. Öffne nun die Zertifikatverwaltung in deinem E-Mail-Programm

- Thunderbird/Linux: „Bearbeiten → Konteneinstellungen → S/MIME → Zertifikate verwalten“

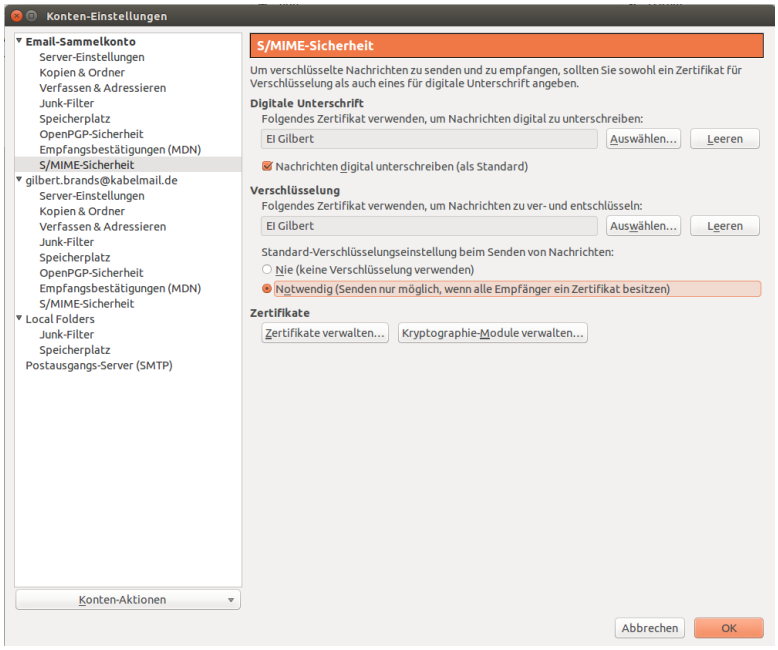
Gehe zunächst in den Bereich „Zertifizierungsstellen“



Über „Importieren“ wählst du „Class 1 …“ aus. Die Verwaltung teilt dir mit, dass sie das Zertifikat importiert hat, und fragt dich noch nach dem Vertrauensniveau. Du solltest E-Mails anklicken



Nun wiederholst du das im Bereich „Ihre Zertifikate“ und gibst nun dein .p12-Zertifikat an. Hier musst du zunächst das Kennwort, dass du beim Export verwendet hast, angeben. Eventuell fragt das System anschließend nach einem Kennwort für die Absicherung des Zertifikatmanagers. Wenn du bereits irgendein Kennwort vergeben hast, musst du es nennen (es kann beispielsweise das Kennwort sein, mit dem du Name/Kennwort-Kombinationen im Browser absicherst), ansonsten lass es ruhig erst einmal frei. Das spart Arbeit. Im letzten Schritt richtest du das Konto so ein, dass immer verschlüsselt und signiert wird:



Jetzt sind wir fertig.

E-Mails schreiben

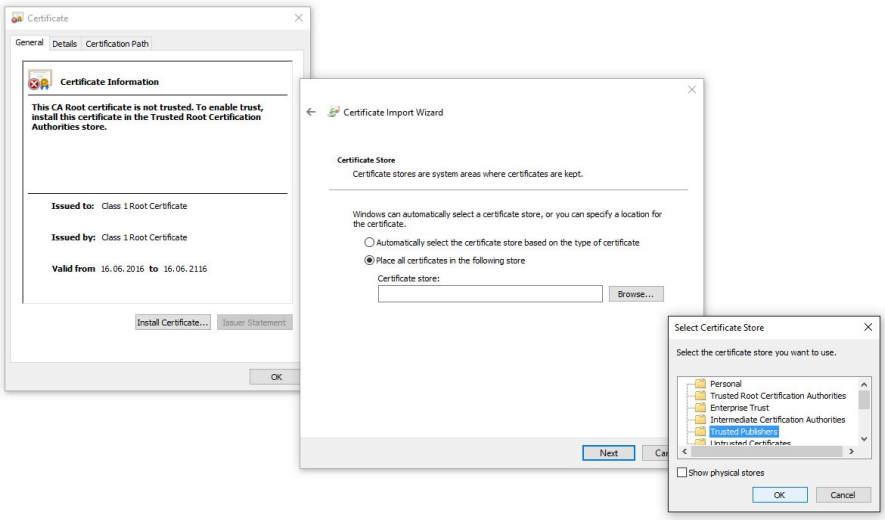
Hier ändert sich für dich nichts. Signieren geht immer, Verschlüsseln aber nur, wenn dein Kollege ein eigenes Zertifikat hat. Falls das nicht der Fall ist, fragt dein Mailer bei dir nach, ob er die Verschlüsselung deaktivieren soll. Dem kannst du zustimmen.

Überrede deinen Kollegen, ebenfalls ein eigenes Zertifikat in dieser Weise zu erzeugen und zu installieren. Sonst braucht du nichts mehr zu tun. Nach der ersten Mail, die noch unverschlüsselt ist, werden alle weiteren automatisch verschlüsselt.

Viel Erfolg! Wenn irgendetwas nicht funktioniert oder du Programme/Betriebssysteme verwendest, die hier nicht beschrieben sind und etwas anders funktionieren, sende mir einfach eine E-Mail: mailto:gilbert@gilbertbrands.de .

Windows und MS Outlook

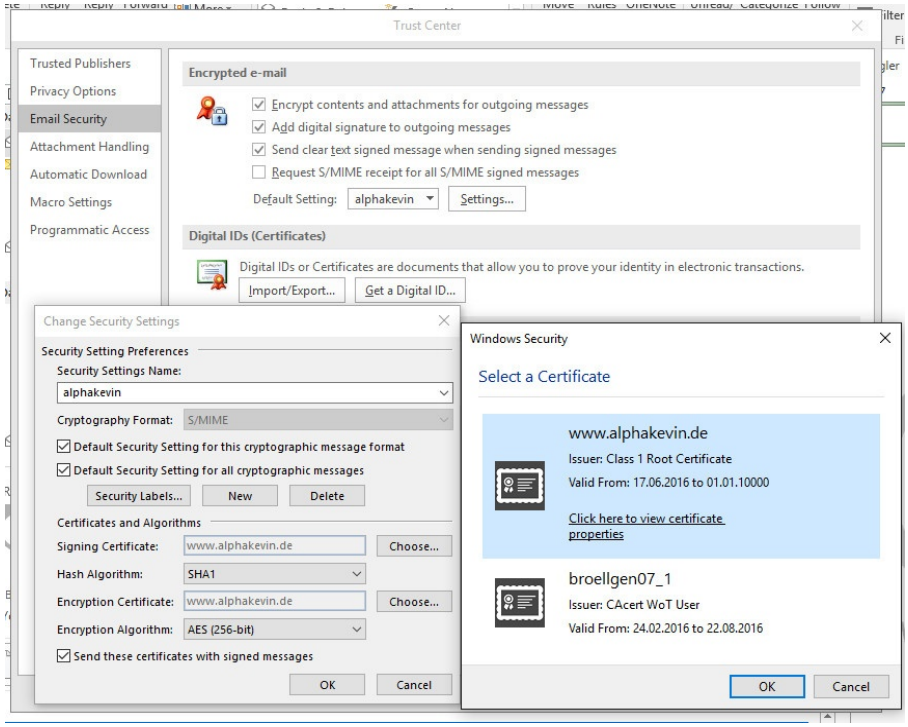
Du benutzt Windows mit MS Outlook statt Thunderbird? Kein Problem! Führe zunächst die Schritte zum Export der Zertifikate wie beschreiben aus. Auch unter Windows ist der Import des Root Zertifikats kinderleicht. Es genügt Doppelklicken auf die Datei Class_1_Electronic_Identity_Root_Certifcate.crt. Es öffnet sich die Zertifikatverwaltung:



Mit dem Klick auf „Install Certificate“ öffnet sich der Import Wizard. Man muss lediglich das Zertifikat unter „Trusted Publishers“ speichern. MS Outlook kennt kurz darauf das Zertifikat.

MS Outlook needs to be configured to use the certificate:

File->Options->Trust Center->Click at trust Center settings... → Email Security:



… und weiter ?

So einfach könnte es sein – wenn die Programme so funktionieren würden, wie sie es angeblich tun sollen. Leider machen weder Thunderbird noch Outlook das, was wir hier eingestellt haben. Software, die nicht funktioniert, und keinen stört es anscheinend.