Public Key Infrastructure: Architektur und Software

Vertrauens-Frage

Digitale Zertifikate verbinden kryptographische Schlüssel mit ihrem Besitzer. Sie sind auf Firmenausweisen, im Web und in E-Mails oder im VPN anzutreffen. Dieser Artikel erklärt die Technologie, beschreibt die Infrastruktur und zeigt, wie man Zertifikate richtig erzeugt und einsetzt. Kay Wondollek

Sie sind die Basis für viele Sicherheitslösungen: digitale Zertifikate und ihre Infrastruktur (PKI, Public Key Infrastructure). Sicher ist ihre Technologie aber nur, wenn sie richtig verstanden und korrekt eingesetzt wird. Bevor dieser Beitrag zur Praxis mit OpenCA kommt, erklärt er daher ausführlich die Grundlagen und Zusammenhänge.

Symmetrisch oder nicht

Die Kryptographie unterscheidet zwischen symmetrischen und asymmetrischen Verfahren. Symmetrische Verschlüsselung setzt denselben Key zum Ver- und die Entschlüsseln ein. Dieser Schlüssel muss streng geheim bleiben, er darf ausschließlich den berechtigten Personen bekannt sein. Problematisch ist dabei der sichere Schlüsseltausch: Um einem Empfänger eine verschlüsselte Nachricht zu schicken, muss der Absender dafür sorgen, dass der Empfänger – und nur dieser – zuvor den gemeinsamen Schlüssel erhält.

Abhilfe schafft die asymmetrische Kryptographie (Public-Key-Verfahren). Sie verwendet ein Schlüsselpaar, bestehend aus einem privaten und einem öffentlichen Schlüssel. Der private muss geheim bleiben, während man den öffentlichen (Public Key) verbreiten darf. Je nach Anwendung (elektronische Signatur oder Verschlüsselung) dient einer der beiden Keys zum Verschlüsseln, der andere zum Entschlüsseln (siehe Kasten „Verschlüsseln und signieren“).

Da der Empfänger nur den öffentlichen Schlüssel des Absenders kennen muss, ist die Kommunikation mit fremden Teilnehmern viel einfacher als mit der symmetrischen Variante. Die Sicherheit der Public-Key-Kryptographie beruht darauf, dass niemand aus dem öffentlichen Schlüssel den privaten bestimmen kann. Eine einfache und leicht verständliche Einführung in die Verfahren liefert [1]. Für das intensive Studium der Thematik ist [2] zu empfehlen.

Ein zentrales Problem bleibt: die Authentizität des öffentlichen Schlüssels. Jeder Teilnehmer muss verlässlich wissen, wessen öffentlichen Schlüssel er gerade benutzt. Er kann sich dazu an eine vertrauenswürdige Instanz wenden, die die Zusammengehörigkeit von öffentlichem Schlüssel und Besitzer bestätigt. Sie setzt dazu Public-Key-Zertifikate ein.

Authentizität des öffentlichen Schlüssels

Public-Key-Zertifikate sind Datenstrukturen, die den Namen des Besitzers (das kann eine Person, ein Rechner, eine URL oder Software sein) mit dessen öffentlichem Schlüssel durch eine elektronische Unterschrift (Signatur) sicher verknüpfen. Der Aussteller des Zertifikats bürgt mit seiner Signatur für die Authentizität des öffentlichen Schlüssels in den Zertifikaten. Aussteller können die Anwender selbst sein (PGP-Modell) oder spezielle, besonders geschützte Autorisierungsstellen, die nach festen Richtlinien verfahren (PKIX-Modell, gesprochen: pikicks, siehe Tabelle 3).

Vertrauenswürdige Instanzen sind üblicherweise Teil einer Zertifikatsinfrastruktur, die neben dem Ausstellen und Signieren der Zertifikate auch für das übrige Zertifikatsmanagement (Veröffentlichung, Rückruf, Erneuerung) zuständig sind. Diese Infrastruktur heißt PKI (Public Key Infrastructure).

Ein bekanntes Beispiel für eine Zertifikatsinfrastruktur, bei der Anwender selbst Zertifikate ausstellen, ist der OpenPGP-Standard, eine von der IETF spezifizierte Adaption von Phil Zimmermanns Pretty Good Privacy (PGP). Im PGP-Modell sind die Benutzer unabhängig von zentralen Autorisierungsstellen. Sie verbreiten ihre Zertifikate durch öf-

fentliche PGP-Keyserver, also über ungesicherte Verzeichnisse, die keinerlei Garantien für die darin veröffentlichten Schlüsseldaten übernehmen. Insbesondere prüfen sie die Zusammengehörigkeit von Schlüssel und Inhaber nicht.

Netz des Vertrauens

Vertrauenswürdig sind PGP-Zertifikate nur, wenn jemand für sie bürgt, der selbst vertrauenswürdig ist. Jeder Teilnehmer kann den öffentlichen Schlüssel eines anderen signieren und so dafür bürgen, dass die Angaben im Schlüsselmaterial korrekt sind. Dadurch entsteht ein Netz des Vertrauens (Web of Trust).

Die Teilnehmer geben ihre öffentlichen Schlüssel und die entsprechenden Zertifikate untereinander weiter. Der Empfänger weist dem Schlüsselzertifikat einen Vertrauensgrad zu und nimmt es in seinen Schlüsselring auf. Der Vertrauensgrad eines PGP-Zertifikats richtet sich nach der Beziehung zwischen Absender und Empfänger. Kennen sie sich persönlich und ist der Empfänger sicher, dass er den korrekten Key erhalten hat, wird er dem Zertifikat vertrauen.

Falls ein Zertifikat von einer vertrauenswürdigen Person beglaubigt (zertifiziert) wurde, kann ihm der Empfänger ebenfalls vollständig vertrauen. Falls leichte Zweifel an der Wahrhaftigkeit eines Zertifikats bestehen, wird er es als eingeschränkt vertrauenswürdig kennzeichnen. Alle übrigen PGP-Zertifikate gelten als nicht vertrauenswürdig.

Damit PGP einen öffentlichen Schlüssel akzeptiert, müssen ein vollständig vertrauenswürdiges oder mindestens zwei eingeschränkt vertrauenswürdige Zertifikate vorliegen. Im Beispiel aus Abbildung 4 kann A den Schlüssel von C erst dann vertrauensvoll anwenden, wenn B den Schlüssel von C zertifiziert oder sobald ein weiteres eingeschränkt vertrauenswürdiges Zertifikat von C vorliegt.

Das hierarchische Modell

Anders als im Web of Trust definiert die PKIX-Working-Group der IETF eine hierarchisch strukturierte PKI, die Internet X.509 Public Key Infrastructure (PKIX). An diesem Rahmenwerk orientiert sich nahezu jede real eingesetzte PKI. Alle Zertifikate werden von einer zentralen Autorisierungsstelle, der Certification Authority (CA), ausgestellt und signiert (Abbildung 5).

Eine PKIX-Implementierung kann mehrere CAs einsetzen, etwa wenn sie sehr viele Zertifikate ausstellen muss (es gibt PKIs mit über einer Million Zertifikaten) oder wegen einer großen räumlichen

Verschlüsseln und signieren

Der übliche Ablauf beim Verschlüsseln ist in Abbildung 1 dargestellt. Eigentlich könnte Absender A dem Empfänger B eine geheime Nachricht N senden, indem er sie mit dem öffentlichen Schlüssel des Empfängers $Ö_B$ verschlüsselt. Als exklusiver Eigentümer wäre nur B in der Lage, mit Hilfe des privaten Schlüssels P_B diese Nachricht wieder zu dechiffrieren.

Hybrid verschlüsseln

Die asymmetrischen Verschlüsselungsverfahren sind jedoch relativ langsam. Deshalb kommen für Nutzdaten meist die schnelleren symmetrischen Algorithmen f_K (etwa DES, IDEA, AES) zum Einsatz. Der Absender verschlüsselt nur den symmetrischen Schlüssel K (oft Sitzungsschlüssel genannt) mit dem öffentlichen Schlüssel des Empfängers und erhält »$crypt_1$«. Das Chiffrat sendet er zusammen mit der symmetrisch verschlüsselten Nachricht (»$crypt_2$«) an den Empfänger. Diese Kombination nennt sich Hybridverfahren.

Elektronische Signatur

Abbildung 2 zeigt das Schema der elektronischen Signatur. Sie bedient sich eines so genannten Hashwerts, den Absender und Empfänger über die zu signierende Nachricht N bilden: h(N). Der Absender verschlüsselt diesen Hash mit seinem eigenen privaten Schlüssel P_A. Hashwerte entstehen durch die kollisionsarme Komprimierung von Daten beliebiger Länge in Daten mit fester, wesentlich kürzerer Länge (typisch sind 128 oder 160 Bit).

Kollisionsarm heißt, dass es praktisch unmöglich ist, zwei unterschiedliche Nachrichten zu finden, die denselben Hashwert ergeben. Ein Hash dient somit als ausreichend eindeutige und wesentlich kürzere Repräsentation der Nachricht. Da nur A den Signierschlüssel P_A kennt, beweist eine gültige Signatur, dass die Nachricht von A unterschrieben wurde. Der Absender übermittelt sowohl die unverschlüsselte Nachricht als auch die Signatur an den Empfänger.

Signatur verifizieren

Beim Verifizieren der Signatur entschlüsselt B mit dem öffentlichen Schlüssel $Ö_A$ des Absenders den Soll-Hashwert der Nachricht h(N). Er vergleicht diesen Wert mit dem Ist-Hashwert, den B selbst über die Nachricht N bildet. Stimmen beide überein, gilt die Signatur als verifiziert und der Urheber der Nachricht ist eindeutig festgestellt. Außerdem ist sichergestellt, dass die Nachricht noch ihren ursprünglichen Inhalt hat.

Derzeit sind als Folge einer EU-Richtlinie zur elektronischen Signatur nationale Gesetzesinitiativen im Gange oder bereits abgeschlossen mit dem Ziel, einen rechtlichen Rahmen für die juristische Anerkennung elektronischer Signaturen bereitzustellen. In Deutschland gibt es seit längerem ein solches Signaturgesetz [3].

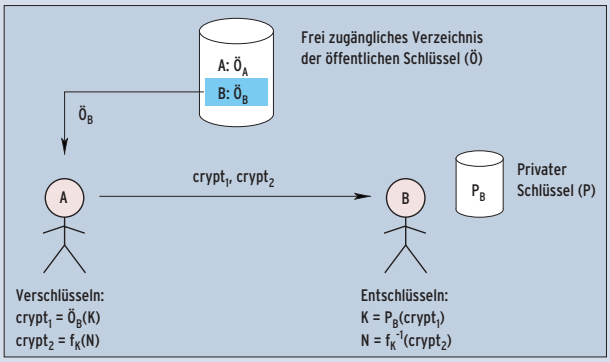

Abbildung 1: Beim Hybridverfahren verschlüsselt Absender A die Nachricht N symmetrisch mit Schlüssel K und dann den Schlüssel K asymmetrisch mit dem öffentlichen Key des Empfängers $Ö_B$. Nur B kann die Nachricht dechiffrieren.

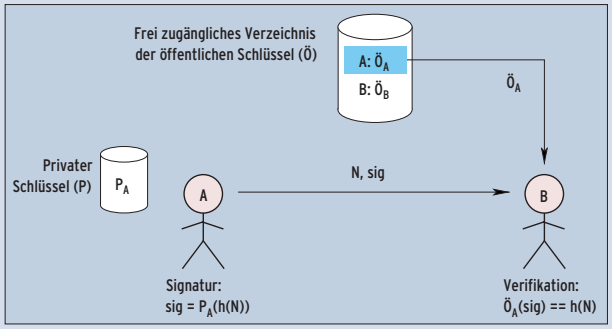

Abbildung 2: Der Absender unterschreibt mit seinem privaten Schlüssel P_A den Hashwert der Nachricht h(N). Nur A kann diese Signatur erzeugen.

Client-Server-Authentifizierung

Auf Basis der elektronischen Signatur (siehe Kasten „Verschlüsseln und signieren“) arbeitet auch die Client-Server-Authentifizierung kryptographischer Protokolle wie SSL/TLS (siehe Abbildung 3). Vereinfacht dargestellt sendet der Server (Server-Authentifizierung) beziehungsweise der Client (Client-Authentifizierung) ein mit dem privaten Schlüssel signiertes Datenobjekt an das Gegenüber. Er beweist damit, dass er im Besitz seines privaten Schlüssels und somit er selbst ist. Von zentraler Bedeutung ist auch hier, dass der private Schlüssel geheim bleibt.

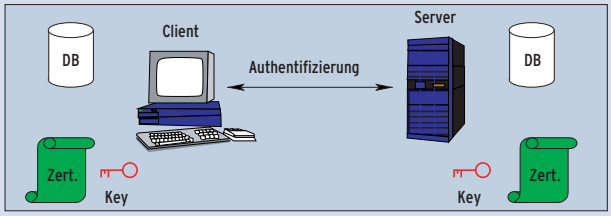

Abbildung 3: In sicheren Protokollen wie SSL/TLS authentifizieren sich Client und Server gegenseitig. Sie benutzen dazu Zertifikate, Datenbanken mit Root-Zertifikaten und ihren geheimen Key.

Gegenseitig authentifizieren

Um die Authentizität des Gegenübers zu prüfen, müssen Client und Server wissen, welcher öffentliche Schlüssel wem gehört. Dazu setzen sie auf PKI. Als Basis ihres Vertrauens benutzen beide

- eine Datenbank mit vertrauenswürdigen Root-Zertifikaten,
- das eigene Zertifikat und den privaten Schlüssel.

Während der gegenseitigen Authentifizierung erfüllt der Client folgende Aufgaben:

- Er erhält und validiert das Server-Zertifikat (eine Zertifikatkette muss bei einem der Root-Zertifikate in der Datenbank des Clients enden),
- authentifiziert den Server und
- sendet sein eigenes Zertifikat sowie
- den Beweis, dass er den passenden privaten Schlüssel besitzt.

Der Server hat ähnliche Aufgaben zu erledigen, aber die Reihenfolge ist anders:

- Er sendet sein Server-Zertifikat sowie
- den Beweis, dass er den passenden privaten Schlüssel besitzt, und
- erhält und validiert das Client-Zertifikat (eine Zertifikatkette muss bei einem Root-Zertifikat in seiner Datenbank enden) und
- authentifiziert den Client.

oder organisatorischen Trennung der Anwender. Die CAs solcher großen PKIs sollten in einer Baumstruktur angeordnet sein. Die Root-CA bildet die Wurzel des Baums, sie stellt Zertifikate für alle Sub-CAs der nächsttieferen Hierarchieebene aus. Diese geben ihrerseits Zertifikate für die nächsten Sub-CAs heraus. Die Sub-CAs der untersten Ebene (die Blätter des Baums) schließlich stellen die Anwender-Zertifikate aus.

In der Praxis sind nur selten mehr als drei Hierarchieebenen erforderlich. Die CA-Zertifikate sind nötig, damit Benutzer die Vertrauenswürdigkeit der CA prüfen können, die ein Anwenderzertifikat ausgestellt hat.

Zertifikatsdepot

Um die ausgestellten Zertifikate öffentlich zugänglich zu machen, hinterlegt sie die CA im so genannten Repository (Abbildung 6), üblicherweise einem LDAP-Server. Die Zertifikate sind durch ihre Signatur vor Fälschungen sicher, das Repository benötigt daher keinen besonderen Schutz.

Mitunter muss eine CA das Schlüsselmaterial selbst erzeugen. Je nach Anwendung ist dies aber nicht zu empfehlen. Um die Urheberschaft einer Signatur nachzuweisen, ist die Beweislage wesentlich eindeutiger, wenn der Anwender sein Schlüsselmaterial selbst erzeugt und den privaten Key nachweislich nie aus der Hand gegeben hat (siehe deutsches Signaturgesetz [3]).

Die CA ist generell für das Management der Zertifikate zuständig. In Szenarien mit vielen, eventuell räumlich weit verteilten Anwendern ist es ratsam, einen

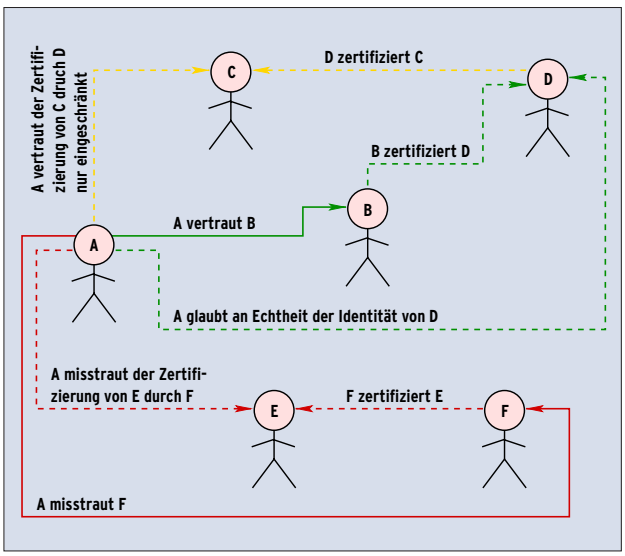

Abbildung 4: Im Web of Trust von PGP richtet sich der Vertrauensgrad eines Zertifikats nach der Beziehung zwischen Absender und Empfänger. A glaubt hier den Identitäten von B und D vollständig, eingeschränkt glaubt er C, aber er misstraut den Identitäten von E und F.

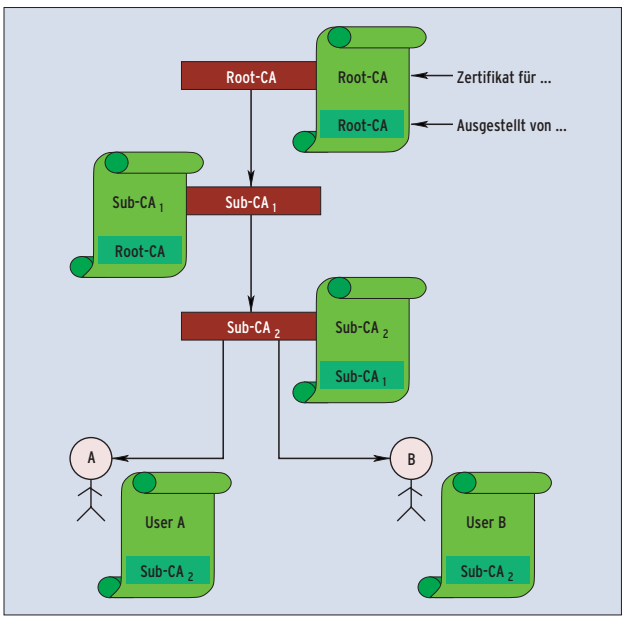

Abbildung 5: Certification Authorities stellen Zertifikate für die nächsttiefere Hierarchieebene aus und erst in der untersten Ebene Anwenderzertifikate.

Teil der Aufgaben an Registration Authorities (RA) vor Ort zu delegieren. Sie kümmern sich um die Aufgaben, bei denen die persönliche Anwesenheit der Anwender notwendig ist: Registrierung, Identifizierung und Authentisierung.

CPS, die Grundlage des Vertrauens

Die Anwender müssen sich darauf verlassen, dass eine CA nur vertrauenswürdige Zertifikate ausstellt. Um dies – zumindest formal – einschätzen zu können, definiert jede PKI ein Certification Practice Statement (CPS). Dort sind die Verfahrensweisen festgelegt, nach denen die PKI Zertifikate ausstellt und verwaltet. Das CPS definiert auch die Pflichten des PKI-Betreibers, die angebotenen Dienste und deren Abwicklung sowie Anforderungen an die PKI-Kunden. Es handelt sich somit um einen Vertrag zwischen PKI-Kunden und PKI-Betreiber (Beispiele siehe RFC 2527 [4]).
Im PKI-Umfeld sind noch zwei weitere Richtlinien anzutreffen. Welche Zertifikate welchem Zweck dienen, ist in der Certificate Policy (CP) festgelegt, siehe ebenfalls RFC 2527. Die PKI-Policy schließlich definiert die Sicherheitsmaßnahmen für den Betrieb einer PKI.

X.509-Zertifikate

Das PKIX-Modell verwendet X.509-Zertifikate (Tabelle 1). Diese enthalten in erster Linie den Namen des Besitzers, dessen öffentlichen Schlüssel sowie die Signatur und den Namen der CA, die das Zertifikat ausgestellt hat. Üblicherweise sind die Namen des Besitzers (Subject) und der herausgebenden CA (Issuer) im Distinguished-Name-Format (DN) angegeben. DNs müssen innerhalb einer PKI eindeutig sein:

```
CN=Myca, OU=TNA, O=Siemens, C=DE
```

Dies könnte der DN für eine CA mit Namen (Common Name) Myca sein, die Zertifikate für die Abteilung (Organisation Unit) TNA innerhalb der Firma (Organisation) Siemens in Deutschland (Country ist DE) ausstellt.
Ähnlich einem Ausweis sollte ein X.509-Zertifikat aus Sicherheitsgründen nur für einen bestimmten Zeitraum gültig sein. Es enthält deshalb eine Gültigkeitsbeschränkung (Validity). Während dieses Zeitraums garantiert die ausstellende CA für die Korrektheit der im Zertifikat beglaubigten Informationen. Noch vor Ablauf der Lebensdauer stellt die CA ein neues Zertifikat aus. Dabei kann, muss aber nicht, zwangsweise das Schlüsselmaterial neu generiert werden.
Die beiden Felder »Issuer Unique ID« und »Subject Unique ID« (ab Version 2) sind optional. Sie sind eine zusätzliche Möglichkeit, um Herausgeber und Besitzer zu identifizieren, werden in der Praxis jedoch so gut wie nie verwendet. X.509-Zertifikate enthalten ab Version 3 (X.509v3) optional eine Liste mit beliebigen Zusatzinformationen, die so genannten Zertifikatserweiterungen (Extensions). Prinzipiell muss eine Anwendung diese Extensions nicht beachten, es sei denn, sie sind als kritisch gekennzeichnet. Kennt eine Anwendung eine kritische Erweiterung nicht, muss sie das Zertifikat zurückweisen.
Einige Extensions reglementieren den erlaubten Einsatz des Zertifikats. Dazu gehören Key Usage sowie Extended Key Usage. Sie reduzieren den Einsatz eines X.509-Zertifikats auf eine bestimmte Anwendung, zum Beispiel auf elektronische Signatur, Verbindlichkeit (Non-Repudiation) oder Verschlüsselung. Andere Extensions unterstützen das Zertifikatsmanagement, indem sie auf den Ort verweisen, an dem wichtige Informationen über gesperrte Zertifikate zu finden sind (CRL Distribution Point, CDP). Wieder andere bieten Alternativen zum DN des Herausgebers oder Besitzers, zum Beispiel dessen E-Mail-Adresse.
Eine genaue Beschreibung des X.509-Formats findet sich in der entsprechenden ITU-T-Recommendation [5] und in RFC 2459 [6].

Gültigkeit prüfen

Bevor sie den öffentlichen Schlüssel eines Zertifikats verwenden, prüfen der Anwender oder seine Anwendung, ob es sich um ein gültiges Zertifikat handelt (Validierung). Folgenden Kriterien müssen erfüllt sein:

- Das Zertifikat selbst ist vertrauenswürdig oder es existiert ein gültiger Zertifizierungspfad zu einem vertrauenswürdigen Zertifikat (Pfadvalidierung).
- Der Gültigkeitszeitraum des Zertifikats hat begonnen und ist noch nicht abgelaufen (Zertifikatslaufzeit).
- Das Zertifikat wurde nicht vorzeitig zurückgerufen (Zertifikatsstatus).
- Die Signatur des Zertifikats ist korrekt und stammt vom Aussteller. ▶

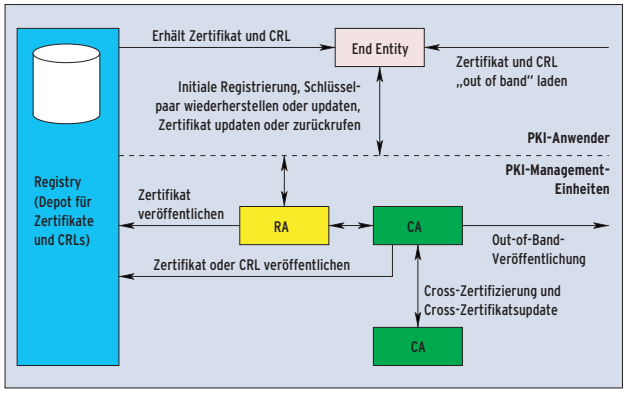

Abbildung 6: In der vollen Ausbaustufe des PKIX-Modells registrieren Registration Authorities (gelb) die Anwender und prüfen ihre Identität. Certification Authorities (grün) stellen Anwenderzertifikate aus und veröffentlichen sie.

Tabelle 1: X.509v3-Zertifikat

Feld	Inhalt
Version	Identifiziert die Version des Zertifikats (etwa v3)
Serial Number	Eindeutige ID des Zertifikats (Integer)
Signature	ID des Algorithmus, mit dem das Zertifikat signiert wurde
Issuer	Eindeutiger Name (DN) des Herausgebers (CA)
Validity	Gültigkeitszeitraum (von ... bis)
Subject	Eindeutiger Name (DN) des Besitzers
Subject Public Key Info	Öffentlicher Schlüssel des Besitzers (und die ID des Algorithmus)
Issuer Unique ID	Eindeutige ID der herausgebenden CA (optional)
Subject Unique ID	Eindeutige ID des Besitzers (optional)
Extensions	Zusatzinformationen (optional), etwa erlaubter Einsatz (»KeyUsage«, »BasicConstraints«)

Der Zertifizierungspfad (Abbildung 7) startet beim Anwenderzertifikat. Dahinter folgen die Zertifikate aller CAs innerhalb des PKI-Baums auf dem Weg zur Root-CA. Root-Zertifikate sind mit dem privaten Schlüssel der Root-CA signiert, die Anwender müssen ihnen daher direkt vertrauen. Eine Root-CA hat keinen weiteren Bürgen, sie selbst ist der Anker des Vertrauens für alle Zertifikate der PKI. Die übrigen Zertifikate des Pfads bürgen für die Authentizität der folgenden Sub-CA-Zertifikate oder des Anwenderzertifikats.

In einem gültigen Zertifizierungspfad muss jedes einzelne Zertifikat gültig sein (entsprechend den genannten Kriterien). CA-Zertifikate werden im X.509-Format durch eine spezielle Zertifikatserweiterung gekennzeichnet (Basic Constraint). Diese legt auch die maximal zulässige Länge des Zertifizierungspfads fest.

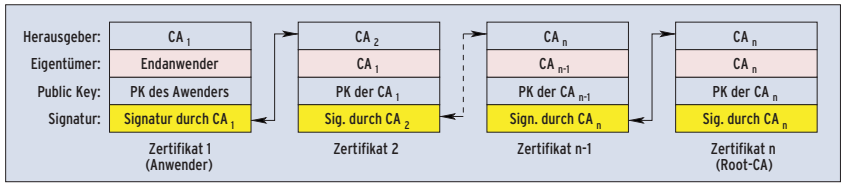

Abbildung 7: Bei einem korrekten Zertifizierungspfad stimmt der Herausgeber eines Zertifikats mit dem Eigentümer des nachfolgenden Zertifikats überein. Nur die Root-CA signiert ihr Zertifikat selbst.

Begrenztes Vertrauen

Um einen Zertifizierungspfad zu validieren, benötigt der Prüfer einen Vorrat an Root-Zertifikaten, denen er vertraut. Viele Anwendungsprogramme (E-Mail-Programme, Browser …) enthalten bereits bei der Auslieferung eine Reihe von Root-Zertifikaten (Abbildung 8).

Die Vertrauenswürdigkeit dieser Zertifikate ist etwas fragwürdig. Sie beruht allein auf der Zuverlässigkeit des Softwareherstellers, der sich hoffentlich von ihrer Authentizität und Gültigkeit überzeugt hat. Auch nachträgliche Manipulationen sind kaum zu bemerken. Außerdem ist es schwierig für den Anwender, auf den Rückruf eines dieser Root-Zertifikate entsprechend zu reagieren. Er weiß in der Regel nicht einmal, welches Root-Zertifikat seine Software gerade zur Pfadvalidierung heranzieht.

Um die Länge der Pfade zu begrenzen, können auch CAs auf tieferen Hierarchieebenen einander Zertifikate ausstellen. Diese Cross-Zertifizierung muss nicht wechselseitig sein. Falls sie erlaubt ist, lassen sich bei der Pfadvalidierung eventuell mehrere gültige Zertifikatspfade konstruieren.

Cross-Zertifizierung ist auch ein probates Mittel, um PKI-übergreifende Vertrauensbeziehungen zu modellieren. Stellen sich die Root-CAs zweier PKIs gegenseitig Cross-Zertifikate aus, existieren für alle Anwenderzertifikate beider PKIs gültige Zertifizierungspfade, ohne dass die Benutzer Root-Zertifikate der jeweils anderen PKI installieren.

Liste mit gesperrten Zertifikaten

Falls ein Zertifikat vor Ablauf seiner Lebensdauer die Vertrauenswürdigkeit verliert, sollte es möglichst umgehend zurückgerufen und gesperrt werden. Das kann nötig werden, wenn trotz aller Vorsicht ein privater Schlüssel in falsche Hände gerät oder wenn ein Mitarbeiter das Unternehmen verlässt. Besonders gravierend ist, wenn der private Schlüssel einer CA kompromittiert wurde. Dann müssen sofort alle Zertifikate, die von der betroffenen CA ausgestellt wurden, zurückgerufen werden.

Manchmal ist es sinnvoll, ein Zertifikat nur zeitweise zu sperren. Verliert ein Mitarbeiter seinen digitalen Firmenausweis (etwa eine Smartcard), lässt sich so der Aufwand für einen neuen Ausweis vermeiden, falls die Chipkarte doch wieder auftaucht.

Neben dem Inhaber eines Zertifikats kann auch der Aussteller ein berechtigtes Interesse an einem Rückruf haben. Angenommen der Nutzer eines Onlinedienstes gibt seinen Schlüssel weiter und ermöglicht so Dritten den kostenlosen Zugang zu diesem Dienst. Der Betreiber wird – sobald er den Betrug erkennt – den Nutzungsvertrag kündigen und den Zugangsschlüssel sperren.

Wird ein Zertifikat gesperrt, müssen die PKI beziehungsweise die zuständige CA dafür sorgen, dass diese Information rechtzeitig allen Beteiligten zur Verfügung steht. Gängige Methode ist es, regelmäßig Zertifikat-Rückruflisten (CRL, Certificate Revocation List) zu veröffentlichen. Üblich sind die von der ISO/ITU Directory Working Group definierten CRLs (Tabelle 2). Die Einträge enthalten die Seriennummern der X.509-Zertifikate und den Zeitpunkt des Rückrufs. Bei Bedarf kann eine CRL Entry Extension auch den Rücknahmegrund nennen.

Jede CRL wird, um die Authentizität der Einträge sicherzustellen, von der zuständigen CA signiert. Eine CA stellt in regelmäßigen Abständen aktualisierte CRLs an öffentlich zugänglichen CRL-Verteilungspunkten (CRL Distribution Point, CDP) bereit. In der Praxis dient dazu meist das Repository (Abbildung 9). Ist der CDP als Extension in den Zertifikaten genannt, weiß der Prüfer bei der Zertifikatsvalidierung, wo er die zugehörige CRL findet.

Das Problem der CRLs ist ihre Aktualität. Der Zertifikatsprüfer ist selbst dafür verantwortlich, immer die aktuelle CRL heranzuziehen. Eine dynamische Online-Statusabfrage hilft ihm dabei, zum Beispiel das von der IETF definierte OCSP (Online Certificate Status Protocol). Der Prüfer sendet während der Validierung eine Anfrage an einen autori-

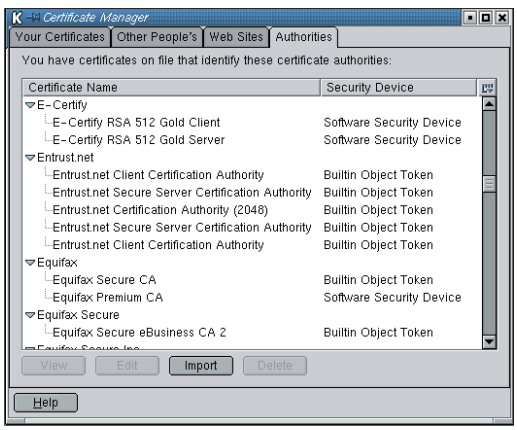

Abbildung 8: Sind in einer Applikation (hier Mozilla) bereits Root-Zertifikate vorinstalliert, sollte der Anwender diesen Zertifikaten nur bedingt vertrauen.

Tabelle 2: Zertifikats-Rückrufliste

Feld	Inhalt
Version	Identifiziert die Version der CRL (etwa v2)
Signature	ID des Algorithmus, mit dem die CRL signiert wurde
Issuer	Eindeutiger Name (DN) des CRL-Herausgebers, meist ist das die CA, die auch die zurückgerufenen Zertifikate herausgegeben hat
This Update	Zeitpunkt (Datum und Uhrzeit), an dem diese CRL erschien
Next Update	Zeitpunkt (Datum und Uhrzeit), an dem die nächste CRL herausgegeben wird
List of revoked Certificates	Die Liste der zurückgerufenen Zertifikate enthält für jedes Zertifikat zwei oder drei Einträge: die Seriennummer des Zertifikats, den Rückruf-Zeitpunkt (Datum und Uhrzeit) sowie optionale Erweiterungen (CRL Entry Extension)
Extensions	CRL-Erweiterungen (optional)

sierten OCSP-Responder. Dieser antwortet mit dem aktuellen Status der gefragten Zertifikate oder liefert eine Fehlermeldung.

X.509-Zertifikate im Eigenbau

Angenommen ein mittelständisches Unternehmen möchte alle Mitarbeiter mit Zertifikaten ausstatten, sie sollen damit ihre E-Mails verschlüsseln. Beim Einführen einer PKI ist es sinnvoll, zunächst nur eine Zertifikatsanwendung umzusetzen, um Erfahrungen zu sammeln und das Vertrauen und die Akzeptanz der Mitarbeiter zu gewinnen.

Am einfachsten wäre es, ein Trustcenter zu beauftragen, das bereits eine etablierte PKI betreibt und passende Zertifizierungsrichtlinien aufweist. Dazu müsste ihm das Unternehmen aber persönliche Daten der Mitarbeiter anvertrauen. Ist das nicht erwünscht, bleibt nur der Weg, eine eigene PKI aufzubauen und zu betreiben.

Der erste Schritt ist es dann, geeignete Richtlinien für die PKI zu definieren, im Einklang mit den Sicherheitsrichtlinien des Unternehmens. Außerdem muss die Firma die Profile der Zertifikate festlegen. Daraus ergeben sich die Anforderungen an die PKI-Architektur.

Bei einem kleineren Unternehmen genügt im Prinzip eine CA, die sowohl Anwenderzertifikate als auch das Root-Zertifikat ausstellt. Nachteil: Wird die CA kompromittiert, muss die Firma alle Zertifikate erneuern und zudem das neue Root-Zertifikat an alle Anwender verteilen. Sinnvoller ist es, zusätzlich eine Root-CA einzusetzen, die ausschließlich das CA-Zertifikat ausstellt.

Die Root-CA braucht nicht ständig verfügbar zu sein. Hat sie das CA-Zertifikat erzeugt, kann man sie abschalten und sicher verwahren. Für das Datenobjekt eignet sich das PKCS#12-Format (siehe Kasten „PKI-Standards“), es besteht aus dem privaten Schlüssel und dem zugehörigen Root-Zertifikat. Abgespeichert und in einem Panzerschrank sicher deponiert entzieht sich die Root-CA jedem Online-Angriff.

Die für Anwenderzertifikate zuständige CA wird zur Sub-CA. Sie veröffentlicht Zertifikate und CRLs im Repository, meist einem LDAP-Server. Ist das Unternehmen auf mehrere Standorte verteilt, sollte es an jedem Standort eine RA einrichten, die auf die nötigten Personaldaten zugreifen kann (Abbildung 9). Die

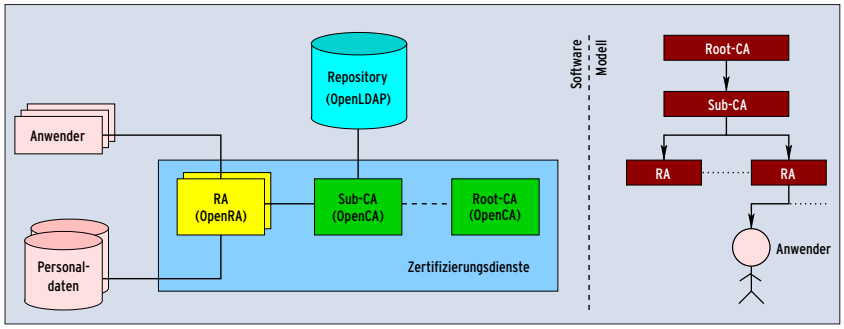

Abbildung 9: Mögliche PKI-Architektur (links) und Vertrauensmodell (rechts) eines kleineren Unternehmens. Wichtig ist die getrennte Root-CA, sie begrenzt den Schaden bei einer Kompromittierung der Sub-CA.

namhaften Hersteller kommerzieller PKI-Produkte unterstützen gegenwärtig jedoch hauptsächlich Windows-Plattformen und bestenfalls Sun Solaris. Unter Linux bleibt die Wahl zwischen einigen Open-Source-Produkten. Für den kommerziellen Einsatz sind dennoch eher die Closed-Source-Produkte zu empfehlen: Sie laufen stabiler, sind weniger fehlerbehaftet und bieten mehr Funktionalität sowie besseren Bedienkomfort.

Open-Source-PKI

Bei den Open-Source-Produkten ist zunächst OpenSSL zu erwähnen **[8]**, die meisten Linux-Distributionen enthalten dieses Paket bereits. OpenSSL ist jedoch kein fertiges PKI-Produkt, sondern eher ein Baukasten mit vielen Kommandozeilentools zum Erstellen und Verwalten von Zertifikaten. Im Rohzustand (ohne Frontend) ist OpenSSL für den produktiven Betrieb einer realen PKI nicht geeignet. Wegen seiner enormen Flexibilität, die das Erstellen von Zertifikaten nahezu jeder erdenklichen Ausprägung erlaubt, ist OpenSSL jedoch hervorragend geeignet für Testanwendungen.

Für den realen Einsatz besser tauglich sind PKI-Produkte mit grafischer Oberfläche, zum Beispiel OpenCA (**[9]**, mit Apache-ähnlicher Lizenz) oder IDX-PKI (**[10]**, GPL). Beide Lösungen sind im Prinzip GUIs für OpenSSL. Weitere PKI-Softwarepakete unter der GPL sind PyCA **[12]** und NewPKI **[13]**. Im Folgenden kommt OpenCA zum Einsatz.

OpenCA

Bevor sich OpenCA (getestet: Version 0.9.1-4) installieren lässt, sind OpenSSL ab Version 0.9.7, Perl ab 5.6.1 sowie der Apache-Webserver inklusive Mod_ssl nötig. Wichtig ist noch OpenLDAP **[14]**. Das Konfigurationsskript warnt leider nicht immer, wenn Softwarekomponenten fehlen. In »doc/OpenCA-guide.pdf« sind nützliche Hinweise zu finden; diese Anleitung ist jedoch unvollständig und die Formulierungen sind teils schwer oder gar nicht verständlich.

Nach dem Auspacken von OpenCA steht »./configure« an, allerdings verlangt das Skript nach einigen Parametern und bricht andernfalls mit einer Fehlermeldung ab (etwa »please specify the hierarchy level«). Da die Liste hierfür länger werden kann, haben die OpenCA-Entwickler im Unterverzeichnis »config« einige Vorlagen abgelegt. Sie sind als Shellskript ausgelegt: das passende auswählen, ins Vaterverzeichnis kopieren, anpassen und starten. Im Suse-Skript fehlt leider der Eintrag »--with-hierarchy-level = ca«.

Um die beiden CAs (Root-CA und Sub-CA), die RAs und das LDAP-Serverinterface zu generieren, muss der Admin die Kommandos »make install-ca«, »make install-ra« und »make install-ldap« auf den jeweiligen Rechnern aufrufen. Die Komponenten landen dabei eventuell an verschiedenen Stellen: Im Configure-Schritt sind vor allem das allgemeine Präfix »--prefix = *Pfad*« und das Verzeichnis für das Webinterface interessant: »--with-httpd-fs-prefix = *Pfad*«.

Das Quellpaket enthält unter »contrib/apache« Vorlagen für die Webserver-Konfiguration. Das Interface der CA darf in einer Produktivumgebung nur lokal zugänglich sein. Auch User sollen die RA kontaktieren, hierfür kommt die Public-Gateway-Schnittstelle zum Einsatz: Anwender beantragen per Browser ein X.509-Zertifikat und fragen über diese Schnittstelle wichtige Zertifikatsinformationen ab, beispielsweise die aktuelle CRL. Das Public Gateway lässt sich mit »make install-pub« installieren.

Beide CAs werden weitgehend identisch initialisiert. Die Root-CA bildet die Vertrauensbasis für alle anderen Zertifikate der PKI, sie ist daher als Erstes aufzusetzen. Admins können hierzu ihren Browser benutzen (Abbildung 10). Während der Initialisierung und Konfiguration zeigt OpenCA übersichtlich die einzelnen Schritte an (Abbildung 11).

PKI-Standards

ISO/ITU-T Directory Working Group: Definiert die Formate für X.509-Zertifikate und Zertifikat-Rückruflisten (CRLs). Insbesondere legt das Gremium den Extensions-Mechanismus sowie die Standarderweiterungen fest und beschreibt, wie X.509-Zertifikate und CRLs in LDAP-Schema-Elemente einzubetten sind (PKI-Repository). [http://www.iso.org/] und [http://www.itu.int/]

IETF PKIX Working Group: Definiert eine auf X.509 basierende Public-Key-Infrastruktur. Die Gruppe veröffentlicht mehrere Internet-Standards, darunter X.509-Zertifikats- und CRL-Profile, legt Protokolle zu Management und Betrieb der PKI fest (LDAP, FTP, HTTP ...) und gibt Leitfäden heraus für Zertifikat-Anwendungsvorschriften und Zertifizierungspraktiken. [http://www.ietf.org/html.charters/pkix-charter.html]

RSA-Labors: Definieren die PKCS (Public Key Cryptography Standards). Anfang 1990 gegründet soll diese Serie die Anwendung von Kryptographie beschleunigen. Derzeit sind 13 PKCS-Standards definiert (#1, #3 und #5 bis #15). Sie sind Teil von vielen formalen und De-facto-Standards wie Ansi X9 und SET (Finanzanwendungen), PKIX, S/MIME und SSL. [http://www.rsasecurity.com/rsalabs/pkcs/] Für PKI-Anwendungen relevant sind:

- PKCS#10: Certification Request Syntax Standard
- PKCS#7: Cryptographic Message Syntax Standard (Import und Export von X.509-Zertifikaten)
- PKCS#12: Personal Information Exchange Syntax Standard (Import und Export öffentlicher und privater Schlüssel sowie von Zertifikaten)

Tabelle 3: Abkürzungen

Ansi	American National Standards Institute	OCSP	Online Certificate Status Protocol
C	Country	OU	Organizational Unit
CA	Certification Authority	PEM	Privacy Enhanced Mail
CDP	CRL Distribution Point	PGP	Pretty Good Privacy
CN	Common Name	PK	Public Key
CP	Certificate Policy	PKCS	Public Key Cryptography Standard
CPS	Certification Practice Statement	PKI	Public Key Infrastructure
CRL	Certificate Revocation List	PKIX	Internet X.509 Public Key Infrastructure
DN	Distinguished Name	RA	Registration Authority
HSM	Hardware Security Module	RBAC	Role Based Access Control
IETF	Internet Engineering Task Force	SET	Secure Electronic Transaction
ISO	International Standards Organization	SSL	Secure Sockets Layer
ITU	International Telecommunications Union	TLS	Transport Layer Security
O	Organization	VPN	Virtuelles privates Netz

Der PKI-Administrator generiert zunächst eine Datenbank für die CA, die später alle CA-spezifischen Daten aufnimmt. Anschließend erzeugt er das Public-Key-Schlüsselpaar (Abbildung 12). Die Software schützt den privaten CA-Schlüssel mit einem Passwort und verpackt den öffentlichen Key in ein X.509-Zertifikat. Hier unterscheidet sich die Initialisierung beider CAs. Die Root-CA erzeugt ein selbst signiertes Zertifikat. Die Sub-CA dagegen exportiert ihren Zertifikatsantrag (auf Diskette) und erhält von der Root-CA das signierte Zertifikat zurück (wieder auf Diskette). Alle Zertifikatsanträge folgen dem PKCS#10-Standard (Kasten „PKI-Standards"). Als Public-Key-Algorithmus steht derzeit nur RSA zur Verfügung.

Der private Schlüssel einer CA, insbesondere der Root-CA-Key, sollte sicher verwahrt werden. OpenCA legt den privaten Schlüssel per Default in der CA-Datenbank ab und zeigt ihn zusätzlich am Bildschirm an, und zwar im PEM-Format (Privacy Enhanced Mail). Besser ist die Anbindung einer Smartcard mit Krypto-Koprozessor: OpenCA nennt das HSM, Hardware Security Module. Dieses Modul erzeugt das Schlüsselpaar der CA und gibt nur den öffentlichen Schlüssel an den Computer weiter. Der private Schlüssel bleibt geschützt in der Hardware der Karte. Damit ist es einfacher, alle geheimen Daten bis zum nächsten Einsatz an einem sicheren Ort zu verwahren. Leider ist für die HSM-Anbindung keine Dokumentation zu finden.

Abbildung 10: Die OpenCA-PKI bringt ein eigenes Webfrontend mit, über das sie sich bequem initialisieren und konfigurieren lässt. Für Admins und Benutzer stehen weitere Oberflächen zur Verfügung.

Abbildung 11: Während der gesamten Initialisierungs- und Konfigurationsphase zeigt OpenCA die notwendigen Schritte. Der Admin muss sie nur in der vorgegebenen Reihenfolge ausführen.

Manager-Account

OpenCA erzeugt zum Abschluss der ersten Initialisierungsphase »CA Management | Initialization | Initialize the Certification Authority« (siehe Abbildungen 10 und 11) die Zertifikatskette für das CA-Zertifikat: »Rebuild CA Chain«. Diese kann der Admin anschließend zusammen mit Informationen für die OpenCA-spezifische rollenbasierte Zugriffskontrolle (RBAC) als so genannte CA-Konfiguration PEM-kodiert auf Diskette exportieren und archivieren. Die CA-Konfiguration der Sub-CA muss er vor Inbetriebnahme an jede ihr zugeordnete RA verteilen.

Während der zweiten Phase der Initialisierung »Create the initial administrator« erzeugt der Administrator für die Root-CA und für die Sub-CA ein Benutzerzertifikat, mit dem er sich später gegenüber der CA authentifiziert. Damit ist die Initialisierung der Root-CA beendet. Um

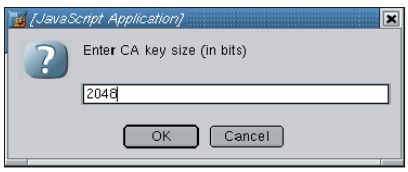

Abbildung 12: Die Schlüsselparameter fragt OpenCA per Javascript-Applikation ab.

die Initialisierung der Sub-CA abzuschließen, generiert der Admin für alle beteiligten RAs Server-Zertifikate.
Nach der Initialisierung folgt die Konfiguration der PKI. Das bedeutet unter anderem, die Rechte aller beteiligten Rollen festzulegen (etwa CA-Operator, RA-Operator und User) und die möglichen Operationen und Funktionen der PKI zu bestimmen sowie neue Module hinzuzufügen (zum Beispiel eine zusätzliche RA). Selbst in der Standardkonfiguration stellt OpenCA jedoch bereits brauchbare Anwenderzertifikate aus.

PKI im laufenden Betrieb

Nachdem die PKI nun einsatzbereit ist, kann sie die Mitarbeiter mit Zertifikaten ausstatten. Der Ablauf dieses Zertifikat-Rollouts ist in Abbildung 13 dargestellt. Zuerst importiert der Anwender das Root-Zertifikat (Schritt 0). Dazu kontaktiert er das Public-Gateway (Abbildung 14) und wählt »Get CA certificate«. Danach beantragt er sein eigenes Zertifikat: »Request a Certificate« (Abbildung 15). In ein Webformular trägt der Anwender im Wesentlichen seinen Namen und seine E-Mail-Adresse ein und wählt aus einer Reihe von spezifischen Rollen (zum Beispiel User, CA/RA-Operator, Mail/VPN/Web-Server …) die passende aus (meist »User«).
Aus der Rolle leitet OpenCA die erforderlichen Zertifikatserweiterungen ab, insbesondere den Anwendungszweck. Leider erlaubt es OpenCA dem Zertifikat-Antragsteller nicht, die Erweiterungen selbst zu definieren. Sie werden stattdessen vom Administrator während der PKI-Konfiguration festgelegt.
Für den Zertifikatsantrag erzeugt der Browser zunächst ein eigenes Schlüsselpaar (Schritt 1 in Abbildung 13). Den privaten Key hält er geheim, aber den öffentlichen Schlüssel sendet er – zusammen mit dem Zertifikatsantrag – an die RA (Schritt 2). Der Administrator der RA prüft die Berechtigung des Zertifizierungsantrags (Schritt 3). Er vergleicht dabei die Einträge mit der Personaldatenbank (ebenfalls Schritt 3), ändert gegebenenfalls die Attribute und leitet dann den geprüften Antrag zusammen mit dem öffentlichen Schlüssel an die Sub-CA weiter (Schritt 4).

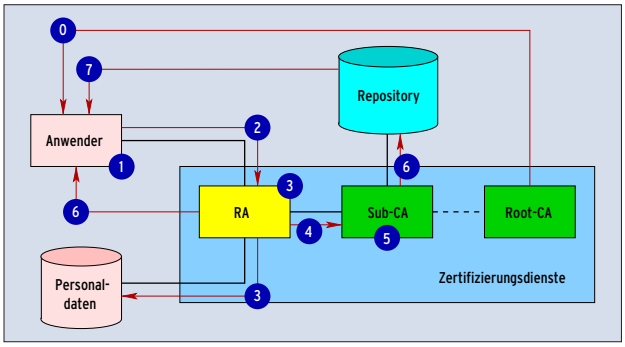

Abbildung 13: Vom Importieren des Zertifikats der Root-CA (Schritt 0) bis zum fertigen Benutzerzertifikat (7) liegen beim Rollout mehrere Zwischenschritte.

Auf den Befehl »Issue Certificate« hin erstellt und signiert die CA das Zertifikat (Schritt 5) und veröffentlicht es anschließend im LDAP-Directory (Schritt 6). Die RA informiert den Anwender automatisch per E-Mail über Ablehnung oder Bewilligung seines Antrags (ebenfalls Schritt 6). Zuletzt importiert der User sein Zertifikat in seine Applikation (Schritt 7). Dabei identifiziert er das Zertifikat mit Hilfe der Seriennummer, die er per E-Mail von der RA erhalten hat.
Soll neben dem Verschlüsseln von E-Mail auch das digitale Signieren wichtiger Dokumente möglich sein, benötigt der Anwender ein zweites Schlüsselpaar und ein zweites Zertifikat. Beim Verschlüsseln von Daten ist es sinnvoll, den privaten Schlüssel sicher zu hinterlegen, andernfalls wäre der verschlüsselte Inhalt verloren, sobald der Key nicht mehr verfügbar ist. Im Gegensatz dazu darf der Anwender bei der digitalen Signatur seinen privaten Schlüssel niemals aus der Hand geben.

Bedingt einsatzbereit

OpenCA ist eine voll funktionsfähige PKI, insbesondere im Sinne des PKIX-Modells. Durch die Weboberfläche ist die CA sehr gut zu bedienen, sowohl für

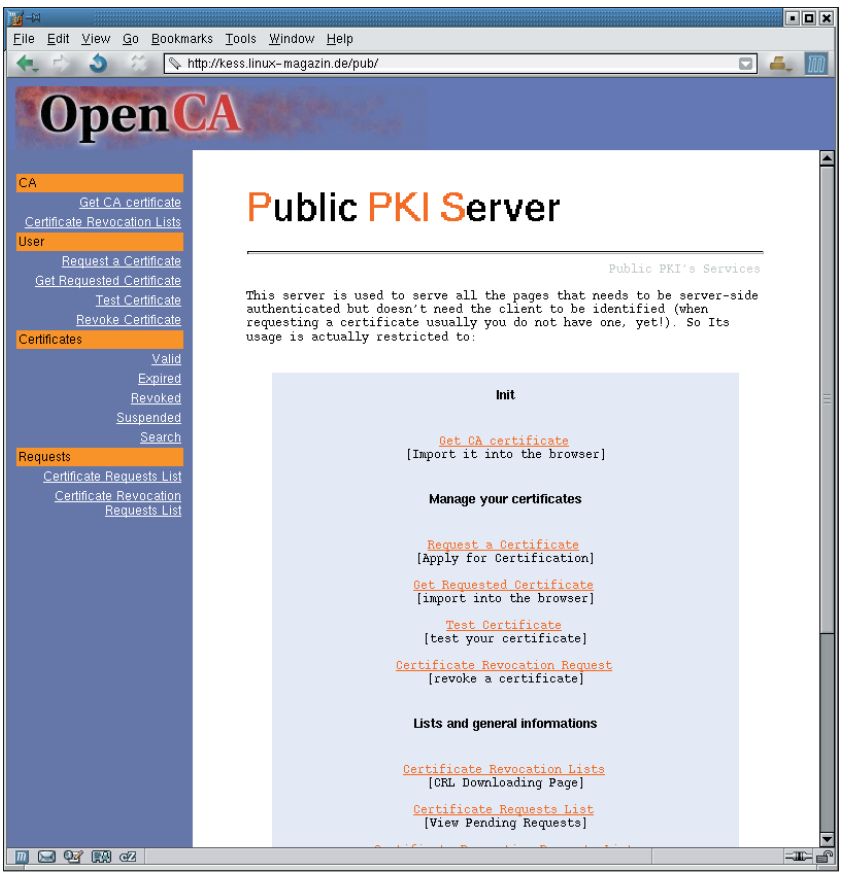

Abbildung 14: Der Anwender importiert das Root-Zertifikat »Get CA certificate« und beantragt anschließend sein Anwender-Zertifikat über das Public Gateway (»Request a Certificate«).

Admins (RA und CA), als auch für Anwender. Dennoch stehen Mängel im Detail und vor allem die lückenhafte Dokumentation einem Einsatz im professionellen Umfeld im Weg. OpenCA befindet sich derzeit aber noch im Entwicklungsstadium, es ist also zu erwarten, dass sich hier vieles bessert. *(fjl)*

Infos

[1] Albrecht Beutelspacher, Jörg Schwenk und Klaus-Dieter Wolfenstetter, „Moderne Verfahren der Kryptographie“: Vieweg-Verlag

[2] Alfred J. Menezes, Paul C. van Oorschot und Scott A. Vanstone, „Handbook of Applied Cryptography“: CRC Press

[3] Deutsches Signaturgesetz (SigG) und Signaturverordnung (SigV): [http://www.bsi.bund.de/esig/basics/legalbas/]

[4] Santosh Chokhani und Warwick Ford, „Internet X.509 PKI Certificate Policy and Certification Practice Framework“, RFC 2527: [http://www.ietf.org/rfc/rfc2527.txt]

[5] ITU-T Recommendation X.509, „Information Technology – Open Systems Interconnection – The Directory: Authentication Framework“

[6] Russell Housley, Warwick Ford, Tim Polk und David Solo, „Internet X.509 Public Key Infrastructure Certificate and CRL Profile“, RFC 2459: [http://www.ietf.org/rfc/rfc2459.txt]

[7] Carlisle Adams und Steve Lloyd, „Understanding the Public-Key Infrastructure – Concepts, Standards, and Deployment Considerations“: New Riders

[8] OpenSSL: [http://www.openssl.org]

[9] OpenCA Research and Development Labs: [http://www.openca.org]

[10] IDX-PKI: [http://idx-pki.idealx.org/index.en.html]

[11] Idealx, Hersteller von IDX-PKI: [http://www.idealx.fr/index.en.html]

[12] PyCA: [http://www.pyca.de/]

[13] NewPKI: [http://www.newpki.org/]

[14] OpenLDAP: [http://www.openldap.org]

Der Autor

Kay Wondollek ist Diplominformatiker und beschäftigt sich seit 1994 beruflich mit dem Thema IT-Sicherheit. Seit 2000 ist er als Senior Consultant im Bereich Information and Communication Networks der Siemens AG tätig. Sein Arbeitsschwerpunkt umfasst das Themengebiet Sicherheitsinfrastrukturen, insbesondere die Themenbereiche Smartcards und PKI.

Abbildung 15: OpenCA lässt dem User die Wahl zwischen mehreren Formaten, in denen er seinen Zertifikatsantrag abgibt. Danach erzeugt der Browser das Schlüsselpaar und gibt den öffentlichen Key an die CA weiter.