

Public-Key-Infrastrukturen

„Policies, Standards, Normen und Vorgaben“

4. Mai 1999

Sommersemester 1999

Betreuer: Markus Ruppert

<http://spellbreaker.org/~chrender/Public-Key-Infrastrukturen>

Zuletzt geändert am 3. Oktober 2005.

Inhaltsverzeichnis

1	Einführung	3
2	Standardisierungs-Organisationen	3
2.1	International Standards Organization	3
2.2	International Telecommunication Union	3
2.3	Institute of Electrical and Electronic Engineers	3
2.4	National Institute of Standards and Technology	3
2.5	Internet Engineering Task Force	3
3	Der X.509-Standard	4
3.1	X.500	4
3.2	Lightweight Directory Access Protocol	6
3.3	ASN.1, BER, DER	7
4	X.509 – Überblick	8
4.1	HTTP/FTP Transfer-Protokolle	11
4.2	Das OCSP-Protokoll	11
4.3	Message- und Protokoll-Formate zur Kommunikation mit einer PKI	12
4.4	Nutzung von LDAP innerhalb der PKI	13
4.5	Policies	13
4.6	Simple Public Key Infrastructure	14
4.7	OpenPGP	15
4.8	Vorgaben der RSA – PKCS	15
4.9	Generic Security Service Application Program Interface	16
4.10	Gesetzliche Vorgaben	17
4.10.1	Signaturgesetz	17
4.10.2	Import- und Exportbestimmungen	17
4.10.3	Deutschland	17
4.10.4	USA	17
4.10.5	Frankreich	18
4.11	E-Mail-PKIs in der Praxis	18
4.11.1	Pretty Good Privacy	18
4.11.2	Privacy Enhanced Mail	19
4.11.3	MOSS	20
4.11.4	S/MIME	20

1 Einführung

Dieser Vortrag beschäftigt sich mit den verschiedenen Standards für Public-Key-Infrastrukturen, die ich bei meiner Suche im World Wide Web gefunden habe. Der erste Teil des Vortrags beschäftigt sich mit den offiziellen Standards für PKIs – von der Internet Engineering Task Force und anderen – der zweite berichtet über (Pseudo-) Standards im Bereich der Electronic Mail.

2 Standardisierungs-Organisationen

2.1 International Standards Organization

An erster Stelle ist eigentlich die ISO zu nennen wenn es um internationale Standardisierung geht. Diese Organisation standardisiert so ziemlich alles, was sich standardisieren läßt und hat bisher ungefähr 11.000 Normen erarbeitet. Zu ihr gehören z.B. das Deutsche Institut für Normung (DIN) und das American National Standards Institute (ANSI). Da die ISO sich allerdings auf keinen Bereich besonders konzentriert, schauen wir uns vier andere Organisationen an, die sich auf Computer und Netzwerke spezialisiert haben.

2.2 International Telecommunication Union

Die International Telecommunications Union (ITU) ist ein Dachverband der Telefongesellschaften der verschiedensten Länder. Diese Organisation hat den Grundstein für den X.509-Standard gelegt, welcher der wichtigste zur Definition von Public-Key-Infrastrukturen darstellt. Die Arbeit an diesem Standard wird heute von der Internet Engineering Task Force fortgeführt (2.4siehe 2.4).

2.3 Institute of Electrical and Electronic Engineers

Die IEEE ist eine weltweite Organisation von Elektroingenieuren, die z.B. mit ihrem 802.3-Standard das Ethernet spezifiziert haben.

2.4 National Institute of Standards and Technology

Das NIST gehört zum US-Handelsministerium. Diese Organisation hat sich unter anderem um die Normung von DES gekümmert.

2.5 Internet Engineering Task Force

Die IETF ist im Gegensatz zur den oben genannten Organisationen keine, die eine feste Mitgliederliste oder feste Konferenzzeiten hat. Die Mitglieder sind ein eher loser Verbund von Wissenschaftlern, Systemanbietern, Netzwerktechnikern und Forschern, die es sich zur Aufgabe gemacht haben Standards für das Internet zu erarbeiten. Die Mitglieder kommunizieren hauptsächlich über E-Mail oder Mailingslisten. Die Arbeitsweise der Gruppe ist folgendermaßen: Soll ein neuer Standard definiert werden, so wird der entsprechende Vorschlag an die IETF geschickt. Sind sich die Mitglieder der Organisation einig, daß dies ein möglicher Kandidat für

einen neuen Standard ist, so wird dieser zum „Internet Draft“ erhoben. Dieser Draft wird von der Internet Engineering Steering Group begutachtet, und nach Durchlaufen dieser Prüfung zum „RFC“, zum „Request for Comment“ erhoben. Nach sechs Monaten werden zwei unabhängige Implementierungen des Standards gesucht, und falls diese existieren, wird der RFC zum Internet Standard erklärt. Verwirrenderweise heißt dieser Standard dann allerdings immer noch RFC. Die IETF ist ziemlich aktiv und hat bisher über 2.000 Standards verabschiedet, die auch öfter überarbeitet werden.

3 Der X.509-Standard

Die Internet Engineering Task Force hat mit X.509 einen internationalen Standard für Public-Key-Infrastrukturen geschaffen. X.509 setzt auf X.500 und LDAP auf und benutzt die „Abstract Syntax Notation One“, kurz ASN.1. Aus diesem Grund folgt hier vor der eigentlichen Erläuterung über X.509 eine kurze Einführung zu diesen beiden Themen.

3.1 X.500

Das Problem der Kommunikation an sich scheint heute größtenteils gelöst. Wenn ich die Anschrift, die Telefonnummer oder die E-Mail-Adresse meiner Kontaktperson kenne, so kann ich in den meisten Fällen problemlos Nachrichten mit ihr austauschen. Es stellt sich also heute eher das Problem, woher ich die nötigen Daten für meine Verbindung bekomme. Falls mein Kommunikationspartner z.B. in einem Land lebt, dessen Sprache ich nicht kenne, so ist es in den meisten Fällen sehr schwer an die Telefonnummer zu kommen. Auch eine E-Mail-Adresse ist im allgemeinen nicht besonders einfach herauszufinden. So ist man auf die Idee gekommen, einen globalen, verteilten Verzeichnisdienst einzurichten. Dieser Dienst ist ähnlich dem Domain Name System, kurz DNS, aufgebaut. Der DNS-Service funktioniert folgendermaßen: Der Benutzer möchte als Beispiel die IP-Adresse zu einem Computernamen herausfinden, so z.B. die Adresse zu `www.tu-darmstadt.de`. Das System des Benutzers richtet dazu zuerst eine Anfrage an einen bekannten DNS-Server. Dieser fragt dann seinerseits beim Server für die `.de`-Domain an, welcher seinerseits die Frage an den Nameserver der TU Darmstadt weiterreicht. Dieser Server kann dann die Frage nach der IP-Adresse des Webservers beantworten. Beim DNS handelt es sich um ein verteiltes System: Es muß keine globale Liste mit allen verfügbaren Computernamen existieren (in früheren Zeiten wurde dies wirklich so gehandhabt). In unserem Fall muß der Nameserver der TU Darmstadt alle Computersysteme registriert haben, die unterhalb der Domain `.tu-darmstadt.de` angesiedelt sind. Der Server der `.de`-Domain muß sich nur um die Einträge direkt unter `.de`-Ebene kümmern usw. Für einen genaueren Einblick in die Funktionsweise von X.500 sei hier auf [STEED99] verwiesen. Einen detaillierteren Überblick bietet die entsprechende RFC – siehe [RFC1309].

Um Daten zu verwalten wird heute normalerweise ein relationales Datenbankmanagementsystem, kurz RDBMS, benutzt. In unserem Fall stellt dies allerdings keine gute Lösung unseres Problems dar: Da die Definitionen für Einträge in der Tabelle immer gleich sind, hat man keine Möglichkeit, die Attribute eines Eintrags zu ändern. Eine Firma möchte aber für ihre Mitarbeiter im Firmenverzeichnis andere Einträge haben wie eine Telefongesellschaft für die Telefonkunden. Abgesehen davon existiert kein standardisiertes Zugriffsprotokoll für diese Datenbanken. Es gibt zwar ODBC und JDBC, diese benötigen aber für jeden Zugriff auf eine bestimmte Datenbank immer noch einen datenbankspezifischen Treiber. Abgesehen davon sind Datenbanken nicht auf das Lesen, also den Zugriff auf die Verzeichniseinträge, optimiert. Da

in unseren Verzeichnissen aber sehr selten geschrieben und hauptsächlich gelesen wird, stellt ein RDBMS auch in diesem Fall keine gute Wahl für das Ablegen unserer Daten dar. X.500 hat also durchaus seine Daseinsberechtigung.

Ein weiterer Vorteil ist der objektorientierte Ansatz von X.500: Jeder Eintrag in einem Verzeichnis kann einer Klasse von Objekten zugeordnet werden. Vererbung ist in diesem Modell ebenfalls erlaubt. So kann z.B. die Eintragsklasse „Firmenchef“ von der Klasse „Firmenange-stellter“ erben. Alle diese Klassen gehören zum sog. Schema eines Verzeichnisdienstes, in dem alle Objektklassen definiert werden. Jeder Eintrag kann verschiedene Attribute besitzen. Manche Attribute – z.B. der Name eines Zertifikateingetümers – müssen gesetzt sein (required), andere – wie etwa die Fax-Nummer des Zertifikatbesitzers – sind nicht nötig, aber zugelassen (allowed). Ein weiterer Vorteil des globalen Systems stellt die eindeutige Benennung jedes Eintrags dar. Jedem Eintrag kann ein global eindeutiger Name zugeordnet werden, der sich aus den „relative distinguished Names“ von jeder Ebene zum absoluten „Distinguished Name“ des Eintrags zusammensetzt (Beispiel siehe Abbildung 1).

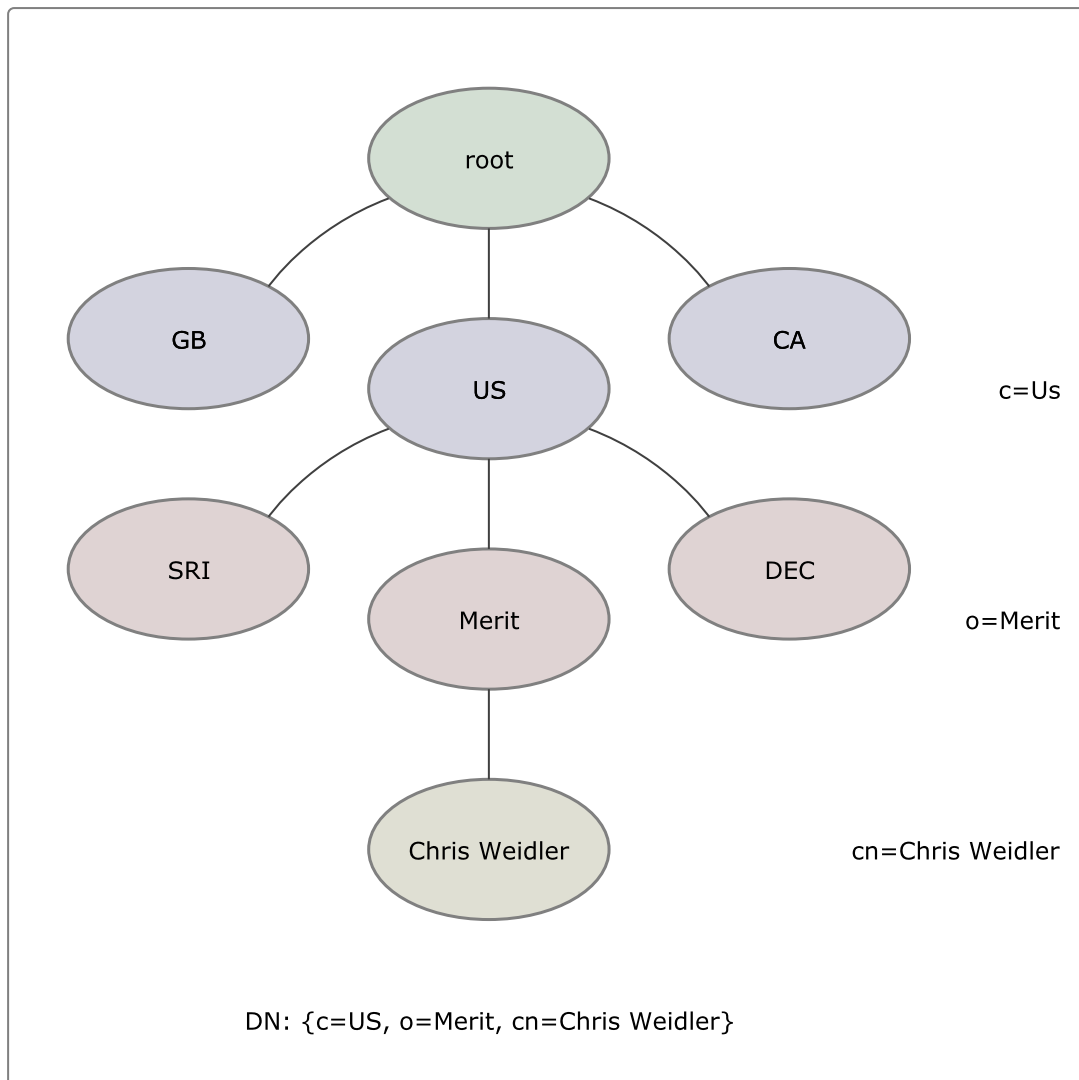


Abbildung 1: Namen in X.500

Nachteile des X.500-Systems sind zum einen die evtl. lange Suchdauer und seine bisher kaum vorhandene Umsetzung. Die Suche kann durch die verteilte Natur des Systems relativ lange

dauern, da in der Kette vom root-Verzeichnisserver bis zum eigentlichen Verzeichniseintrag sehr viele Rechner vorhanden sein können. Das andere Problem ist die Verbreitung: X.500-Systeme sind bisher kaum anzutreffen. Das Deutsche Forschungsnetz (DFN) unterhält ein X.400-System, den Vorgänger von X.500.

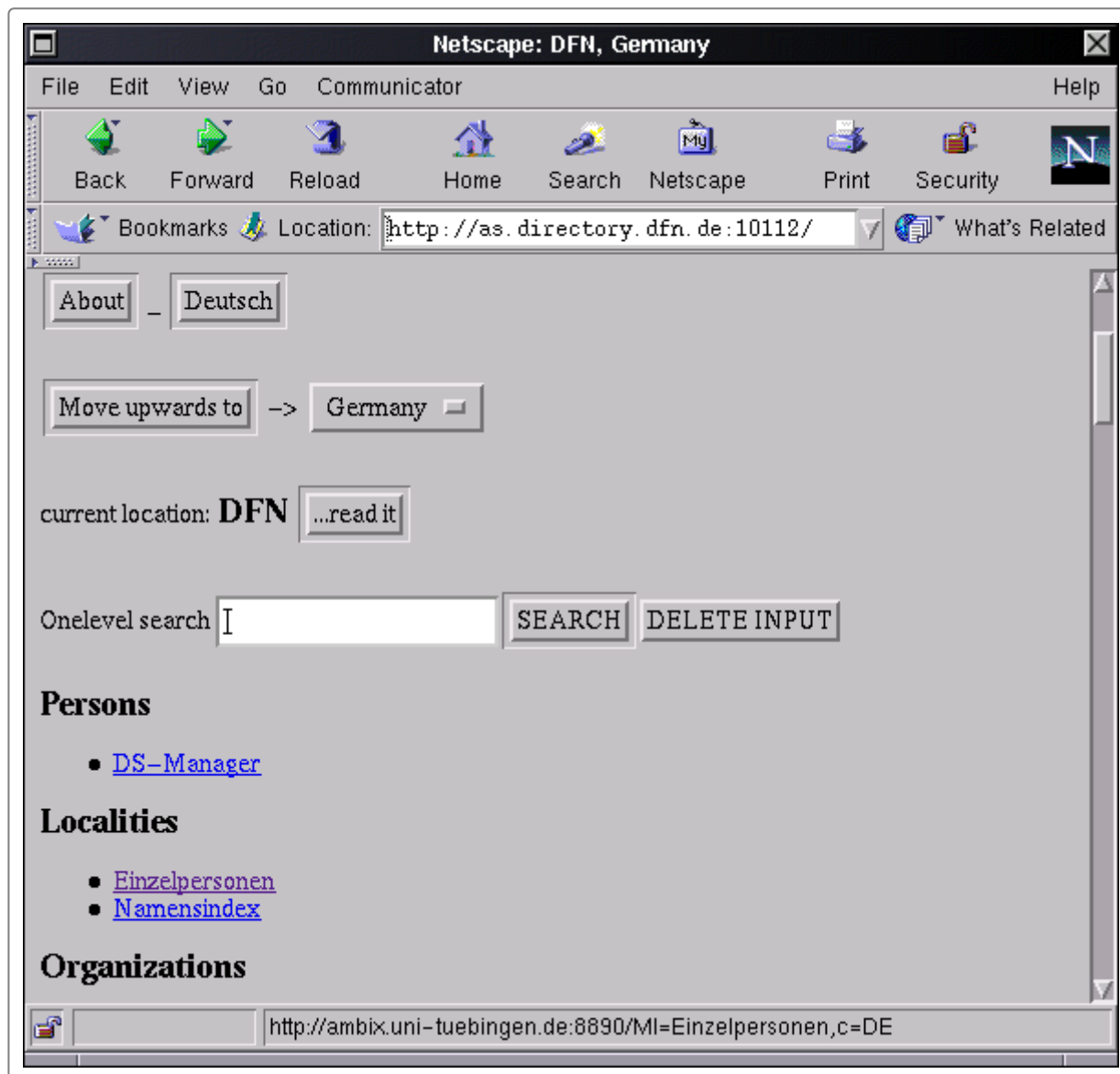


Abbildung 2: Verzeichnisdienst beim DFN

3.2 Lightweight Directory Access Protocol

Das Lightweight Directory Access Protocol dient zum Zugriff auf X.500-Verzeichnisse. Es wird in [RFC2251] spezifiziert. Es soll Read- und Updatefunktionalität implementieren, momentan ist allerdings nur die Read-Funktionalität komplett spezifiziert. Aus diesem Grund nennt sich das Protokoll auch „Lightweight“ – es ist noch nicht vollständig. Vorteil gegenüber dem „normalen“, für X.500 gedachten Directory Access Protocol: Zumindest der Lesezugriff ist soweit definiert, daß man eine brauchbare Anwendung schreiben kann, ohne später Probleme mit noch nicht spezifizierten Dingen zu haben. Trotzdem gibt es schon jetzt die verschiedensten Erweiterungen (definiert in [LDAPEXT]). Freie Implementierungen sind inzwischen ebenfalls verfügbar (siehe [KIR99]).

3.3 ASN.1, BER, DER

Die Abstract Syntax Notation One ist eine Methode, um abstrakte Objekte zu spezifizieren. Abstrakt bedeutet hier implementierungsunabhängig, man möchte nur das System spezifizieren und überläßt die exakte Implementierung dem Programmierer. Da mit ASN nur Datenobjekte und kein Code spezifiziert werden, ist das System relativ einfach zu verstehen. Ein detaillierter Einblick findet sich in [KAL93].

Die einfachen Datentypen umfassen folgende:

- Bitstring, einfache Aneinanderreihung von 0 und 1.
- IA5String, ein ASCII-String.
- Integer, ein Element aus den natürlichen Zahlen.
- NULL, der Nullwert.
- Object Identifier – stellt eine Referenz zu einem anderen Objekt dar.
- Octet String
- Printable String, ein String mit druckbaren Zeichen.
- T61String, ein String aus T.61 (Acht-Bit-)Zeichen.
- UTCTime, ein Wert der „coordinated Universal Time“ oder auch „Greenwich Mean Time“.

Beispiele für strukturierte Typen – „structured Types“:

- Sequence, eine geordnete Menge von einem oder mehr Elementen
- Sequence Of, eine geordnete Menge aus null oder mehr Elementen
- Set, eine ungeordnete Menge von einem oder mehr Elementen
- Set Of, eine ungeordnete Menge aus null oder mehr Elementen
- Choice, ein Element aus einer angegebenen Liste
- ...

Hier ein Beispiel für die Nutzung der ASN.1-Syntax. Die folgenden Zeilen spezifizieren die Gültigkeitsdauer des Zertifikates. Grün markierte Wörter sind Objekt Identifier, violette Schlüsselwörter.

```
Validity ::= SEQUENCE {
    notBefore      Time,
    notAfter       Time }

Time ::= CHOICE {
    utcTime        UTCTime,
    generalTime    GeneralizedTime }
```

Die mit der ASN.1 beschriebenen Objekte möchte man natürlich auch abspeichern oder über das Netzwerk versenden. Dazu muß man sie binär kodieren – also in einen Bitstring konvertieren. Dazu existieren zwei standardisierte Vorgehensweisen – die „Basic Encoding Rules“ (BER) und die „Distinguished Encoding Rules“ (DER). Die Basic Encoding Rules haben den Nachteil, daß sie ein ASN-Objekt auf mehrere gültige Bitstrings abbilden können. Die Distinguished Encoding Rules dagegen bilden ein Objekt auf genau einen Bitstring ab. In den meisten Standards wird die Anwendung von DER bevorzugt.

4 X.509 – Überblick

Auszug aus dem Internet Draft [X509CH] zu X.509:

„Many Internet protocols and applications which use the Internet employ public-key technology for security purposes and require a public-key infrastructure (PKI) to securely manage public keys for widely-distributed users or systems. The X.509 standard constitutes a widely-accepted basis for such an infrastructure, defining data formats and procedures related to distribution of public keys via certificates digitally signed by certification authorities (CAs). RFC 1422 specified the basis of an X.509-based PKI, targeted primarily at satisfying the needs of Internet Privacy Enhanced Mail (PEM). Since RFC 1422 was issued, application requirements for an Internet PKI have broadened tremendously, and the capabilities of X.509 have advanced with the development of standards defining the X.509 version 3 certificate and version 2 certificate revocation list (CRL).“

Angefangen hat alles mit der X.509-Spezifikation der ITU-T. Darin wurde zum ersten Mal alle nötigen Voraussetzungen für eine PKI definiert, zusammen mit den nötigen Datenformaten und Protokollen. Allerdings war die erste X.509 etwas oberflächlich formuliert – viele Felder oder ihre Inhalte waren nicht besonders genau definiert. In der zweiten Version des Standards wurden dem Format des Zertifikats zwei neue Felder hinzugefügt, die den Zugriff auf Daten innerhalb von Verzeichnissen regeln. Als 1993 Privacy Enhanced Mail spezifiziert wurde, erkannte man einige Mankos innerhalb der X.509. Die PKIX-Workgroup erstellte darauf hin Version 3 der X.509-Spezifikation, die jetzt den Standard für eine PKI festlegt. Dabei wurden einige Zugeständnisse gemacht, nachdem man einige Schwächen in den vorherigen Versionen des Standards ausgemacht hatte. Am bemerkenswertesten ist die Tatsache, daß jetzt Felder hinzugefügt werden dürfen, und zwar nach Belieben der PKI-Betreiber. Man ist also nicht gezwungen, sich explizit an die Vorgaben zu halten. Wenn man ein neues Feld benötigt, welches nicht spezifiziert worden ist, so kann man dieses einfügen. Dies macht natürlich die Interoperabilität zwischen unterschiedlichen PKIs deutlich komplizierter, es hat sich allerdings in der Praxis herausgestellt, daß dies eine notwendige Voraussetzung ist.

Wie oben schon beschrieben, nutzt X.509 LDAP (v2) zum Zugriff auf die PKI-Daten, welche in einem LDAP-Verzeichnis abgelegt sind.

Eine PKI besteht aus fünf verschiedenen Komponenten:

- Certification Authorities (CAs), die Zertifikate ausgeben und zurückrufen
 - Organizational Registration Authorities (ORAs), die die Verbindung zwischen Zertifikat und Zertifikatseigentümer bestätigen können
-

- Certificate holders – Eigentümer von Zertifikaten, die Zertifikate ausgestellt bekommen und Dokumente digital signieren können
- Clients, die in der Lage sind, digitale Signaturen und den Zertifikationspfad eines Public Keys einer CA zu überprüfen
- Repositories – Datenspeicher, die Zertifikate und Certificate Revocation Lists (CRLs) verfügbar machen.

Wie sieht denn nun ein Zertifikat aus? [RFC2459]

```

Certificate ::= SEQUENCE {
    tbsCertificate      TBSCertificate,
    signatureAlgorithm  AlgorithmIdentifier,
    signatureValue      BIT STRING }

TBSCertificate ::= SEQUENCE {
    version             [0] EXPLICIT Version DEFAULT v1,
    serialNumber        CertificateSerialNumber,
    signature            AlgorithmIdentifier,
    issuer              Name,
    validity            Validity,
    subject             Name,
    subjectPublicKeyInfo SubjectPublicKeyInfo,
    issuerUniqueID      [1] IMPLICIT UniqueIdentifier OPTIONAL,
                      -- If present, version shall be v2 or v3
    subjectUniqueID     [2] IMPLICIT UniqueIdentifier OPTIONAL,
                      -- If present, version shall be v2 or v3
    extensions          [3] EXPLICIT Extensions OPTIONAL
                      -- If present, version shall be v3
}

Version ::= INTEGER { v1(0), v2(1), v3(2) }

CertificateSerialNumber ::= INTEGER

Validity ::= SEQUENCE {
    notBefore          Time,
    notAfter           Time }

Time ::= CHOICE {
    utcTime            UTCTime,
    generalTime        GeneralizedTime }

UniqueIdentifier ::= BIT STRING

SubjectPublicKeyInfo ::= SEQUENCE {

```

```
algorithm      AlgorithmIdentifer,  
subjectPublicKey  BIT STRING }
```

```
Extensions ::= SEQUENCE SIZE (1..MAX) OF Extension
```

```
Extension ::= SEQUENCE {  
    extnID      OBJECT IDENTIFIER,  
    critical    BOOLEAN DEFAULT FALSE,  
    extnValue   OCTET STRING }
```

Es folgt eine minimale Liste von Funktionen, die eine PKI erfüllen muß. In einigen Fällen bietet eine PKI evtl. noch weitere Funktionen an.

- Registration
Der Benutzer meldet sich bei der CA, von der er ein Zertifikat erhalten will, an. Dazu muß er seinen Namen – echten Namen, IP Adresse o.ä. – und einige andere Attribute angeben, die im Zertifikat gespeichert werden. Danach ist es Aufgabe der CA (evtl. mit Hilfe der RA), diese Angaben zu überprüfen.
 - Initalization
Die Initalisierung findet statt, wenn der Benutzer oder das Client System die Daten erhalten, um mit der PKI zu kommunizieren. Dabei erhält der Client evtl. sofort sein Zertifikat, oder zuerst das Zertifikat der CA.
 - Certification
In diesem Prozeß wird der Public Key des Clients zertifiziert. Der Client bekommt das Zertifikat zugesandt oder muß es selbst aus dem Repository herabladen.
 - Key Pair Recovery
Einige Implementationen werden so angelegt sein, daß ein Backup des Private Keys erstellt wird oder Daten abgelegt werden, die den Private Key wiederherstellbar machen.
 - Key Generation
Das Private/Public-Key-Paar kann entweder vom Client selber oder von der CA erstellt werden.
 - Key Update
Alle Schlüsselpaare müssen nach einem vorher festgelegten Zeitraum erneuert werden. Das kann aber auch schon früher geschehen, z.B. wenn man der Ansicht ist, daß der eigene Private Key von jemand anderem gelesen wurde. Dies kann auch mit dem Private Key der rootCA geschehen. In einem solchen Fall kann kein von dieser rootCA zertifizierter Schlüssel mehr genutzt werden, bis alle Schlüssel neu zertifiziert wurden.
 - Cross-Certification
Wenn zwei verschiedene PKI-Systeme untereinander sicher kommunizieren wollen und ihre bestehenden Schlüssel weiterverwenden möchten, so ist eine Cross-Zertifizierung nötig. In diesem Fall wird einer CA von einer anderen CA ein Zertifikat ausgestellt. Dabei wird ein Public Key ausgegeben, der mit dem Private Key zum Signieren von Zertifikaten zusammenhängt.
-

Cross-Zertifizierung muß nicht zwangsläufig in beide Richtungen laufen: Wenn CA1 CA2 zertifiziert, aber nicht umgekehrt, so kann ein Client, der ein CA1-Zertifikat besitzt, Nachrichten von einem CA2-User empfangen, aber nicht umgekehrt.

- **Revokation**
Falls ein Zertifikat vor dem Ende seiner ursprünglichen Gültigkeitsdauer nicht mehr verwendet werden darf, wird das Zertifikat in eine Certificate Revocation List eingetragen, die von der CA zu erhalten ist. Das Problem bei dieser Art von Management ist, daß eine neue CRL eigentlich zu bestimmten, periodischen Zeitpunkten verfügbar gemacht wird. Dadurch kann es vorkommen, daß ein Zertifikat ungültig ist, aber noch nicht in die aktuelle CRL eingetragen wurde. Wird eine neue CRL früher verfügbar, so müßte dies allen CA-Benutzern mitgeteilt werden, was auch zu Problemen führen kann (wenn der Client z.B. grade nicht online ist).
- **Certificate and Revocation Notice Distribution/Publication**
Die PKI ist verantwortlich dafür, daß Zertifikate und Certificate Revocation Notices im System verteilt werden.

4.1 HTTP/FTP Transfer-Protokolle

Da Directory Services noch nicht weit genug verbreitet sind, benutzt man FTP/HTTP um an sein Zertifikat zu kommen oder um es zu verbereiten (siehe [RFC2585]). Die Verwendung dieser Prokolle setzt natürlich voraus, daß man genug über den Benutzer oder das Zertifikat weiß, um den Ablageort zu kennen. Mit Hilfe der Directory Services stellt sich dieses Problem natürlich nicht. Zum Transfer der Zeritifikate sind zwei MIME-Typen reserviert worden:

- application/pkix-cert
- application/pkix-crl

Es wird empfohlen, daß Files, die Zertifikate enthalten, die Endung `.cer` erhalten. Jedes `.cer`-File enthält genau ein Zertifikat, welches im DER-Format kodiert ist. Entsprechend enthalten Certificate Revokation List-Files die Endung `.crl`. Jedes CRL-File enthält genau eine CRL, ebenfalls im DER-Format kodiert.

Da Zertifikate und CRLs digital signiert sind, sind keine weiteren Sicherheitsvorkehrungen nötig. Weder Zertifikate noch CRL brauchen geheim gehalten zu werden. Das einzige Problem, welches sich bei der Verwendung von HTTP ergeben könnte, ist die Verwendung von Cache-Proxies. Manche Proxies implementieren den Check auf eine neuere als die gecachte vorhandene Version eines Files nicht korrekt, und so könnte ein veraltetes File an dem Empfänger gelangen.

4.2 Das OCSP-Protokoll

Der Zweck des OCSP-Protokolls besteht darin, den aktuellen Status eines Zertifikates zu erfahren, ohne CRLs zu benutzen. Das Protokoll wird in [RFC2560] definiert. Eine OCSP-Anfrage besteht aus den folgenden Daten:

- Protocol Version
 - Service Request
-

- Target Certificate Identifier
- Optionale Erweiterungen die vom OCSP Responder weiterverarbeitet werden können

Der OCSP-Responder checkt nach eingegangener korrekter Anfrage, ob er in der Lage ist, den angefragten Service zu leisten. Wenn ja, gibt er das entsprechende Ergebnis zurück. Die Antwort sollte digital signiert sein.

Eine Antwort hat das folgende Format:

- Version der Antwort-Syntax
- Name des Antwortenden
- Antworten für jedes Zertifikat. Diese sind folgendermaßen definiert:
 - Target Certificate Identifier(?)
 - Status des Zertifikats. Mögliche Werte: Gültig, Zurückgewiesen, Unbekannt.
 - Zeitliches Interval, in dem die Antwort gültig ist
 - Optionale Erweiterungen
- Optionale Erweiterungen
- Signatur-Algorithmus OID(?)
- Über den Hashwert der Antwort berechnete Signatur

4.3 Message- und Protokoll-Formate zur Kommunikation mit einer PKI

Mit Hilfe dieser speziellen Messages – definiert in [CMMF] kann ein Client:

- Die Zertifizierung eines Public Keys anfordern
- Um Revokation eines Keys bitten
- Direkt Änderungen von einer PKI gesendet bekommen. Etwa ein Update der CA-Keys, neuere Zertifikate oder neue CRLs
- Daten der PKI abfragen, etwa das eigene Zertifikat oder das eines Anderen, sowie CRL herunterladen

In [RFC2797] wird außerdem ein „Certificate Management Protocol“ (CMS) vorgeschlagen. Damit soll ein Interface definiert werden, welches mit CMS- und PKCS 10-Produkten umgehen kann (auch S/MIMEv3 certificate enrollment protocol).

4.4 Nutzung von LDAP innerhalb der PKI

In [LDAPV2] wird ein minimales Schema definiert, welches es ermöglichen soll, PKIX in einer LDAPv2-Umgebung zu nutzen. Es werden primär End-Entities und Certification Authorities spezifiziert.

Spezifikation der End-Entities:

```

pkiUser OBJECT-CLASS ::= {
    SUBCLASS OF { top}
    KIND auxiliary
    MAY CONTAIN {userCertificate}
    --ID { joint-iso-ccitt(2) ds(5) objectClass(6) pkiUser(21)}

userCertificate ATTRIBUTE ::= {
    WITH SYNTAX Certificate
    EQUALITY MATCHING RULE certificateExactMatch
    ID joint-iso-ccitt(2) ds(5) attributeType(4) userCertificate(36) }

```

Spezifikation der Certification Authorities:

```

pkiCA OBJECT-CLASS ::= {
    SUBCLASS OF { top}
    KIND auxiliary
    MAY CONTAIN {cACertificate |
                 certificateRevocationList |
                 authorityRevocationList |
                 crossCertificatePair }
    --ID { joint-iso-ccitt(2) ds(5) objectClass(6) pkiCA(22)}

```

4.5 Policies

Zertifizierungspolitik: Legt genau fest, wer unter welchen Umständen ein Zertifikat erhält und wann ein Zertifikat gesperrt wird. Für PKIX werden in [RFC2527] nur die wichtigsten Punkte beschrieben.

“According to X.509, a certificate policy is ‘a named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements.’ ... The certificate policy concept is an outgrowth of the policy statement concept developed for Internet Privacy Enhanced Mail ... For example, a particular certificate policy might indicate applicability of a

type of certificate to the authentication of electronic data interchange transactions for the trading of goods within a given price range.

A certificate policy, which needs to be recognized by both the issuer and user of a certificate, is represented in a certificate by a unique, registered Object Identifier. The registration process follows the procedures specified in ISO/IEC and ITU standards.”

Polices sind vor allem wichtig, wenn es an die Cross-Zertifizierung von Certification Authorities geht. Betrachten wir z.B. Benutzer A, dessen Zertifikat laut Policy zum Versenden oder Empfangen von E-Mail gedacht ist, sowie Benutzer B, dessen Zertifikat für eCommerce geeignet ist – es sagt aus, daß Benutzer B mindestens 18 Jahre alt ist und daß ein einzelner Geschäftsabschluß bis 1000,— DM möglich ist. Wenn nun die verschiedenen Certification Authorities der Benutzer A und B eine Cross-Zertifizierung vornehmen wollen, so müssen sie ihre Policies vergleichen. Die CA des Benutzers A hat kein Problem, die Zertifikate der anderen CA anzuerkennen – Anwender, die über 18 sind und Geschäfte bis 1000,— DM abschließen dürfen, können auch E-Mail versenden und empfangen. Dagegen ist der umgekehrte Weg nicht so einfach zu gehen: Allein die Tatsache, daß ein Anwender E-Mail versendet heißt noch lange nicht, daß dieser mindestens 18 Jahre alt ist und Geschäftsabschlüsse tätigen darf.

Als Beispiel seien hier einige Punkte aus der Policy für E-Mail-Zertifikate des Deutschen Forschungsnetzes genannt.

- Da die Verzeichnisdienste nach Möglichkeit in ein globales Verzeichnis integriert werden sollten, ist eine der nötigen Angaben der Distinguished Name, unter dem das Verzeichnis zu finden ist. Beim DFN ist dies { g=dfnpca, ou=pca, p=dfn, a=d400, c=de }.
- Es werden Zertifikate für PGP, PEM, SSL und S/MIME ausgestellt.
- Der private Key der CA und evtl. andere private Keys, die zur CA selbst gehören werden auf einem isolierten Rechner gespeichert. Isoliert bedeutet in diesem Fall, daß keine Netzwerkverbindung nach außen besteht. Der Datentransfer erfolgt per Floppy oder Magnetband. Die minimale Schlüssellänge beträgt 2048 Bit.
- Die Anwender müssen ihre Schlüssel auf Smartcards speichern. Steht eine solche nicht zur Verfügung, so kann der private Key alternativ auch auf der Festplatte abgelegt werden – in diesem Fall muß der Schlüssel allerdings durch eine beliebige Art von Activation Data (Paßwort, PIN) zusätzlich geschützt werden. Die minimale Schlüssellänge für den Anwender ist 1024 Bit.

4.6 Simple Public Key Infrastructure

Das Problem bei einer Public-Key-Infrastruktur nach dem X.500-Standard besteht darin, daß viele Firmen und Organisationen, die ihre Daten in das Verzeichnissystem einbringen sollten, ihre Mitarbeiterlisten als wertvolles Gut betrachten und sie nicht so ohne weiteres freigeben. Es gutes Beispiel dafür sind Mitarbeiterlisten eines Geheimdienstes. Es ist also nicht zu erwarten, daß sich X.500 als globales Verzeichnissystem durchsetzt.

Im Gegensatz zu „herkömmlichen“ Public-Key-Infrastrukturen beschäftigt sich die Simple Public Key Infrastructure eher mit Authorisation als mit Authentifikation. Das System funktioniert auf folgende Art und Weise: Namen werden nicht global, sondern lokal betrachtet. Lokale

Namen sind die Namen, mit denen wir Leute anreden, die wir in unsere Notizbücher schreiben und mit denen wir uns auf bestimmte Personen beziehen. So kann eine Person von zwei anderen durch völlig verschiedene Namen bezeichnet werden. Alle Namen sind also lokal in Bezug auf eine bestimmte Person, die diese Namen verwendet. Eine Beispiel für eine solche Benennung wäre z.B.:

```
george: (name fred)
```

Dieser Ausdruck repräsentiert ein einfaches Feld mit dem Namen „fred“ in dem von george definierten Namensraum. Wenn nun fred wiederum einen Namen definiert, z.B.

```
fred: (name sam)
```

so kann george sich auf dieses Feld beziehen, indem er folgenden Ausdruck verwendet:

```
george: (name fred sam)
```

Mit dieser Verknüpfung von Namen kann man allerdings noch nicht viel anfangen. Man benötigt für viele Zwecke immer noch einen globalen Identifizierer. Dieser kann z.B. durch den Public Key oder – falls eine kollisionsfreie Hashfunktion vorliegt – durch dessen Hashwert gegeben sein, da der Public Key global eindeutig sein sollte. Wenn z.B. ein Name außerhalb eines Zertifikates benutzt wird, so benötigt man eine globale Referenz für diesen Namen. Dieser Name ist der Public Key gefolgt von einem oder mehreren Namen relativ zum Schlüsselinhaber.

```
(name (hash sha1 —TLCgPLFlGTzgUbcaYLW8kGTEnUk=—) jim therese)
```

4.7 OpenPGP

OpenPGP, spezifiziert in [RFC2440], ist auf der Basis von PGP Version 5.x aufgebaut. OpenPGP definiert ein einheitliches Nachrichtenformat, so daß Clients, die auf dieser Spezifikation aufbauen, Nachrichten untereinander austauschen können. Es existieren keine großen Unterschiede zu den bisherigen PGP-Versionen, da es auch grade die Absicht darstellt, beide so kompatibel wie möglich zueinander zu halten.

Da OpenPGP auf PGP 5.x aufbaut, muß die Zusammenarbeit mit älteren Versionen speziell gehandhabt werden. Dies wird in speziellen RFCs definiert. In [IMC] wird zum Verschlüsseln TripleDES (DES EDE3 Eccentric CFB), zum Signieren ElGamal mit DSS und als Hashalgorithmus SHA-1 vorgeschlagen.

4.8 Vorgaben der RSA – PKCS

Die RSA-Laboratories haben mit den Public-Key Cryptography Standards (siehe [PKCS]) verschiedene Grundlagen für die Public-Key-Verschlüsselung und die dazugehörigen Infrastrukturen geschaffen. Die im Jahre 1991 veröffentlichten Dokumente wurden von Repräsentanten von Apple, Digital, Lotus, Microsoft, MIT, Northern Telecom, Novell und Sun in Verbindung mit RSA Laboratories geschaffen. Mit pkcs werden solche Dinge wie der RSA-Algorithmus, Zertifikats-Syntax-Erweiterungen, Syntax für die Zertifikatsabfrage und ähnliches formalisiert. Seit seiner Einführung sind verschiedene pkcs-Standards in den unterschiedlichsten Entwicklungen verwendet worden, z.B. in Privacy-Enhanced Mail oder Apple's PowerTalk. Obwohl die pkcs von keinem öffentlichen Gremium spezifiziert oder abgesegnet wurden, sind sie ziemlich weit verbreitet.

4.9 Generic Security Service Application Program Interface

Das Generic Security Service Application Program Interface bietet eine grundlegende Funktionalität zum Einbinden von Verschlüsselungstechnik in Programme und erlaubt es so, Programme einfacher auf anderen Plattformen zu portieren.

Eine vollständige Beschreibung ist hier sicher fehl am Platz. Es folgt ein kurzes Beispiel für einen Verbindungsaufbau. Für weitere Informationen sei auf [RFC2078] verwiesen.

Das Programm von Alice, welches eine sichere Verbindung zu Bob öffnen möchte, ruft zuerst `GSS_Acquire_cred()` mit einigen Informationen über die Verbindung, z.B. der geplanten „Lebensdauer“ der Verbindung auf. Diese Funktion liefert ein sog. Credential („Credential A“), welches an die Funktion `GSS_Init_sec_context()` weitergereicht wird. Die Funktion liefert ein sog. Token („Output Token A“), welches an Bobs System weitergereicht werden kann. Nur Tokens werden zur Kommunikation zwischen Rechnern eingesetzt.

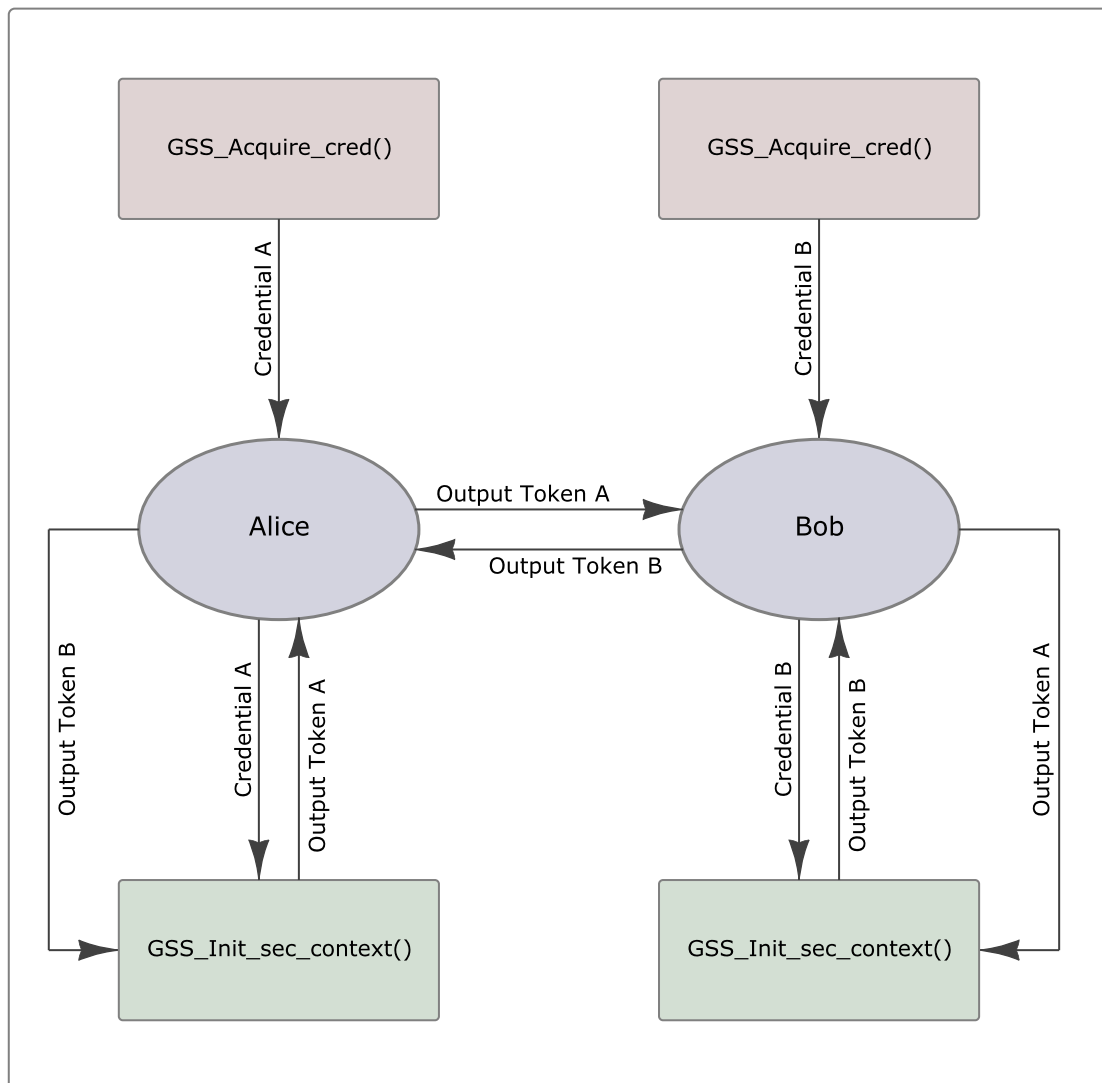


Abbildung 3: Übersicht über einen GSS-API-Verbindungsaufbau

Bobs System holt sich ebenfalls ein Credential von der Funktion `GSS_Acquire_cred()` („Credential B“) und liefert dieses zusammen mit dem Output Token A von Alices System an die Funktion `GSS_Accept_sec_context()`. Das sich daraus ergebende Token („Output Token B“)

wird an Alices System zurückgeschickt. Dort wird das erhaltene Token wieder an die Funktion `GSS_Init_sec_context` übergeben. Damit ist die Verbindung initialisiert, und beide Parteien können über die gesicherte Verbindung kommunizieren.

4.10 Gesetzliche Vorgaben

Überraschenderweise sind momentan nicht besonders viele gesetzliche Vorgaben zu Public-Key-Infrastrukturen in Kraft. Neben dem noch relativ bekannten Signaturgesetz existieren gesetzliche Grundlagen in Amerika nur in wenigen Staaten, z.B. in Utah. Genauere Informationen finden sich auf den „PKILaw“- oder „Crypto Law Survey“ Webseiten (siehe [PKILAW] bzw. [LAWSRV]).

4.10.1 Signaturgesetz

Die Bundesregierung möchte auf privatwirtschaftlicher Basis eine bundesweite Infrastruktur für die Zuordnung von Signaturschlüsseln zu natürlichen Personen einführen. Die Regierung stellt dabei nur die „root-CA“ zur Verfügung und zertifiziert weitere Trustcenter, die dann im Gegenzug Personen oder andere CAs bzw. Organisationen zertifizieren können. Die Bestimmungen sind inzwischen per Gesetz festgelegt und auch online verfügbar (siehe [BSILIT]).

Der wichtigste Unterschied zur X.509-Spezifikation besteht darin, daß der private Schlüssel stets in persönlichem Gewahrsam gehalten werden muß. Das bedeutet, daß der Private Key nicht auf der Festplatte gespeichert sein darf. Man muß in auf einer SmartCard oder einer Diskette mit sich führen können. Außerdem muß im Zertifikat die Policy eingetragen sein, für die das Zertifikat gilt.

4.10.2 Import- und Exportbestimmungen

4.10.3 Deutschland

Deutschland ist am Wassenaar-Abkommen beteiligt. Es umfaßt vor allem die folgenden interessanten Punkte:

- Alle Krypto-Produkte bis zu 56 Bits Schlüssellänge sind frei für den Export
- Kryptosoftware für den Massenmarkt und Hardware bis 64 Bits sind frei für den Export
- Der Export anderer Kryptoprodukte bedarf (weiterhin) einer Lizenz
- Freie Software ist von Kontrollen völlig ausgeschlossen

In Deutschland selbst existieren zur Zeit keine weiteren Beschränkungen zum Import/Export von Kryptoprodukten.

4.10.4 USA

Es existieren keine Importbeschränkungen für Kryptoprodukte. Beim Export sieht es dagegen deutlich anders aus: Es dürfen nur Produkte mit einer maximalen Schlüsselgröße von 56 Bit exportiert werden, und auch das nur nach einem Review und dann nur für sechs Monate. „Data Recovery Crypto“, also Verfahren, bei denen die Regierung der USA Zugriff auf Plaintext oder Schlüssel erhält, dürfen auch mit größeren Schlüsseln exportiert werden.

4.10.5 Frankreich

Frankreich war lange Zeit noch weiter eingeschränkt als viele andere Staaten: Die maximale Größe für die Schlüssel betrug 40 Bit. Größere Schlüssel durften nur für private Zwecke verwendet werden, und auch dann nur bis zu einer maximalen Länge von 128 Bit. Die Gesetze, die momentan in der Vorbereitung sind, zielen allerdings auf eine deutliche Liberalisierung ab, evtl. werden die Beschränkungen weitestgehend aufgehoben.

4.11 E-Mail-PKIs in der Praxis

4.11.1 Pretty Good Privacy

PGP hat sich trotz seines etwas komplizierten Trust-Verfahrens (siehe weiter unten) erstaunlich schnell verbreitet, was evtl. auch daran liegt, daß keine zentrale CA benötigt wird, um Zertifikate bzw. Schlüsselpaare zu erstellen – der Benutzer kann dies selbst vornehmen. PGP ist inzwischen in der Version 6 verfügbar.



PGP basiert auf dem sogenannten „Web of Trust“. Um festzustellen, ob man einer anderen Person vertrauen kann, die man nicht kennt, betrachtet man das Vertrauen, welches eine bekannte Person der unbekannteren entgegenbringt. Angenommen, Alice möchte den öffentlichen Schlüssel von Bob zu ihren anderen öffentlichen Schlüsseln hinzufügen. Dabei hat sie mehrere „Vertrauenslevel“ zu Auswahl:

- Completely trusted – wenn ein anderer Schlüssel von Bob's neuen Schlüssel signiert wird, dann kann dieser neue Schlüssel ebenfalls zu Alice's bestehenden hinzugefügt werden.
- Marginally trusted – Ein Schlüssel, der von Bob's Key signiert wurde, muß noch von einem anderen signiert worden sein, damit er in Alice's Sammlung aufgenommen werden kann.
- Untrusted – Bob's Schlüssel wird nicht benutzt um zu bestimmen, ob andere Keys zu Alice's Schlüsselbund hinzugefügt werden.
- Unknown – Diesem Schlüssel kann kein Vertrauenslevel zugeordnet werden.

Dadurch, daß die User ihre Schlüsselpaare selber erstellen, müssen sie auch selbst für ihre Verbreitung sorgen. Dieses Austauschen der Schlüssel darf dann aber nicht via E-Mail laufen. Um nun an Schlüssel von Personen zu gelangen, die man nicht kennt, muß man anderen Personen vertrauen. Dies bedeutet aber, daß man, um andere Schlüssel per E-Mail und nicht über einen sicheren Kanal zu bekommen, Leuten vertrauen muß, die man noch nie getroffen hat. Dadurch, daß man praktisch seine eigene CA ist und alle Schlüsselupdates selber erledigen muß, ist die Verwendung von PGP in größerem Rahmen nicht praktikabel. Für kleinere Gruppen ist dieses Verfahren jedoch noch gut zu überschauen, weswegen PGP inzwischen relativ bekannt und verbreitet geworden ist.

4.11.2 Privacy Enhanced Mail

Privacy Enhanced Mail („PEM“) wurde 1993 als Internetstandard für verschlüsselte E-Mail entworfen. Es war das Ziel, E-Mail mit Vertraulichkeit, Authentizität und Integrität zu versehen.

Die PEM Public-Key-Infrastruktur ist streng hierarchisch aufgebaut. Der „root“-Knoten ist die Internet Policy Registration Authority („IRPA“), die globale Zertifikations-Policies für das gesamte Internet entwirft. Die IRPA zertifiziert nur Policy Certification Authorities („PCAs“). Jeder der PCAs entwirft spezifischere Policies und zertifiziert wiederum CAs, die auf den Richtlinien der PCAs aufbauen. Die CAs können nun direkt PEM-Zertifikate ausstellen oder weitere CAs zertifizieren.

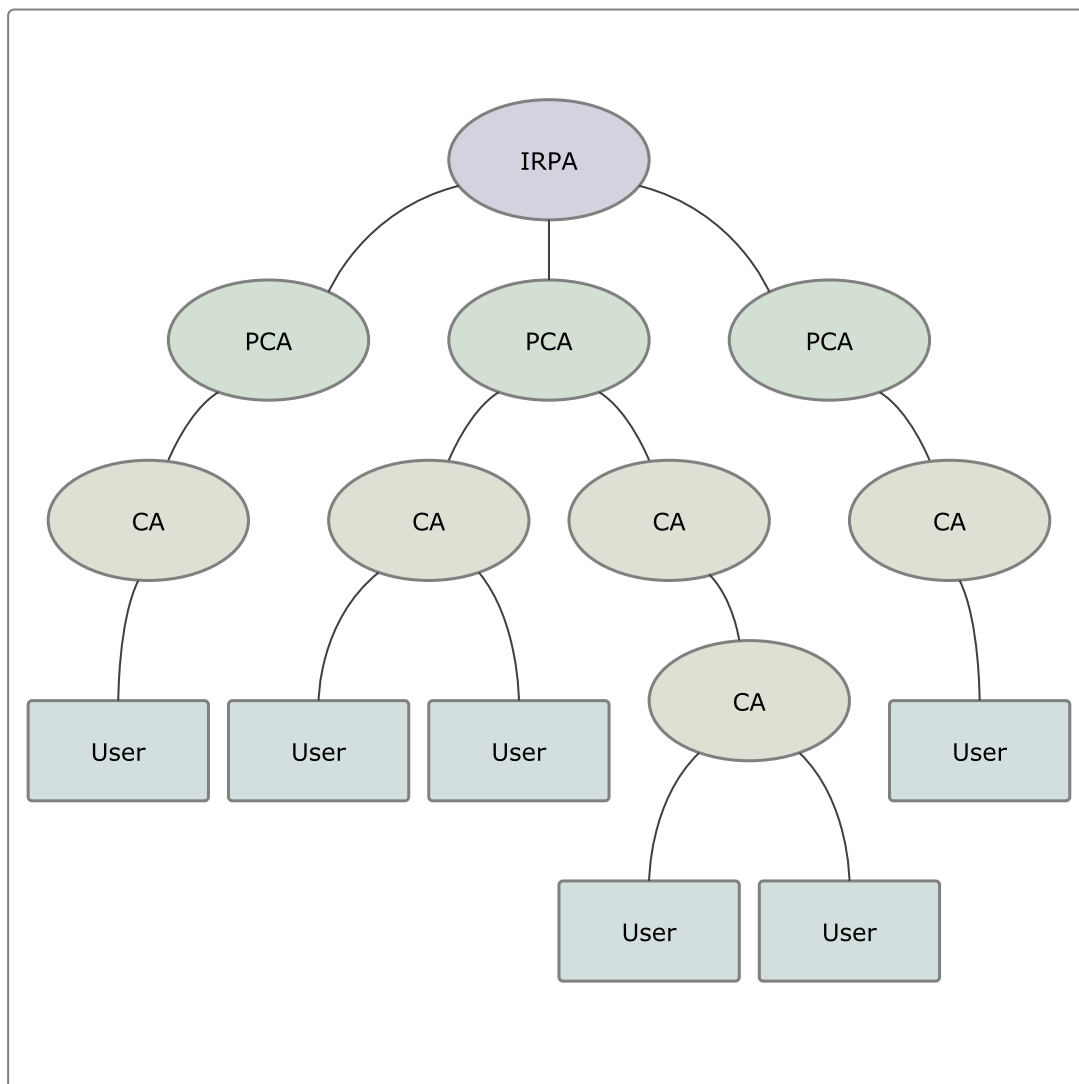


Abbildung 5: PEM-Zertifizierungsstruktur

In der Praxis hat sich leider erwiesen, daß der hierarchische Aufbau der PEM-PKI gut bei entsprechend strukturierten Organisationen funktionierte, wie z.B. großen Firmen, die ebenfalls eine Hierarchie besitzen. Der größte Teil des Internets baut jedoch auf einer Peer-to-Peer-Struktur auf und ist deswegen nicht so einfach in einen verwaltungstechnischen Rahmen zu fassen. Alle Benutzer müssen außerdem der globalen rootCA, der IRPA, vertrauen. Ein weiteres

Problem besteht darin, daß PEM keinen automatischen Validierungsmechanismus kennt, und die Policies jeder CA dem Benutzer bereits bekannt sein müssen.

PEM baut auf X.509v1 auf und besitzt eine streng hierarchische CA-Struktur. Es wird die X.509v1 CRL-Struktur benutzt; CRLs werden per E-Mail zu den Usern gebracht. Der User muß zuerst irgendwie an den Key der IRPA kommen, danach können alle Managementoperationen über sichere Kanäle abgewickelt werden. Eine Besonderheit bei PEM stellt der Anonymitätsmechanismus dar: sog. „Persona CAs“ stellen Zertifikate für Personen aus, die nicht garantiert identifiziert werden können.

4.11.3 MOSS

MOSS (MIME Object Security Services) erlaubt die Benutzung von PEM mit MIME-Attachments. MOSS muß sich eher als eine Rahmenvorgabe anstatt einer genauen Spezifikation vorstellen. Es gibt viele mögliche Implementationen, so daß zwei korrekt implementierte MOSS-Mail-Programme sich evtl. gar nicht verstehen.

4.11.4 S/MIME

S/MIME – definiert in [SMIME] ist ein weiterer Ansatz zur Erweiterung von E-Mail mit Verschlüsselung und Authentifikation. Die Nachricht wird mit einer symmetrischen Chiffre verschlüsselt, und ein Public-Key-Algorithmus wird für den Schlüsselaustausch und die Signaturen verwendet. Drei verschiedene symmetrische Chiffren werden empfohlen: DES, Triple-DES und RC2. RSA wird als Public-Key-Chiffre eingesetzt.



Falls eine signierte E-Mail verschickt wird, so existieren zwei Möglichkeiten die Signatur in der Nachricht unterzubringen: Entweder als Attachment oder direkt in der Nachricht selber in einer binären Form. Probleme treten auf, wenn der Empfänger im ersten Fall S/MIME verwendet, die Nachricht von einem Mailserver aber in einer normalerweise unproblematischen Art leicht verändert wurde – z.B. Zeilen anders umgebrochen oder abschließendes Whitespace entfernt. In diesem Fall kommt das empfangende Mailprogramm zu dem Schluß, daß die E-Mail manipuliert worden ist. Ein anderes Problem tritt auf, wenn der Empfänger keinen S/MIME-Client benutzt und die Nachricht in binärer Form erhält. Diese ist dann natürlich nur noch schwer lesbar.

Literatur

- [STEED99] Douglas Steedman: „x500 Directory Services“, Technology Appraisals, ISBN: 1871802245
- [RFC1309] C. Weider; J. Reynolds und S. Heker: „Technical Overview of Directory Services“
<http://www.ietf.org/rfc/rfc1309.txt>
- [RFC2251] M. Wahl; T. Howes und S. Kille: „Lightweight Directory Access Protocol (v3)“
<http://www.ietf.org/rfc/rfc2251.txt>
- [LDAPEXT] The Internet Engineering Task Force: „LDAP Extensions“
<http://www.ietf.org/html.charters/ldapext-charter.html>
- [KIR99] Christian Kirsch und Ingo Lütkebohle: „... daß ich Rumpelstielzchen heiß“, Heise Zeitschriften Verlag
<http://www.heise.de/kiosk/archiv/ix/1999/5/155>
- [KAL93] Burton S. Kaliski Jr.: „A. Layman’s Guide to a Subset of ASN.1, BER and DER“, RSA Laboratories
<ftp://ftp.rsa.com/pub/pkcs/ascii/layman.asc>
- [X509CH] Stephen Kent und Warwick Ford: „Public-Key Infrastructure (X.509) (pkix)“
<http://www.ietf.cnri.reston.va.us/html.charters/pkix-charter.html>
- [RFC2459] R. Housley; W. Ford; W. Polk und D. Solo: „Internet X.509 Public Key Infrastructure; Certificate and CRL Profile“
<http://www.ietf.org/rfc/rfc2459.txt>
- [RFC2585] R. Housley und P. Hoffman: „Internet X.509 Public Key Infrastructure“
<http://www.ietf.org/rfc/rfc2585.txt>
- [RFC2560] Michael Myers; Rich Ankney; Ambarish Malpani; Slava Galperin und Carlisle Adams: „X.509 Internet Public Key Infrastructure“
<http://www.ietf.org/rfc/rfc2560.txt>
- [CMMF] C. Adams und M. Myers: „Internet X.509 Public Key Infrastructure“
- [RFC2797] Michael Myers; Xiaoyi Liu; Jim Schaad und Jeff Weinstein: „Certificate Management Messages over CMS“
<http://www.ietf.org/rfc/rfc2797.txt>
- [LDAPV2] Sharon Boeyen; Tim Howes und Pat Richard: „Internet X.509 Public Key Infrastructure“
<http://www.ietf.org/rfc/rfc2587.txt>
- [RFC2527] S. Chokhani und W. Ford: „Internet X.509 Public Key Infrastructure“
<http://www.ietf.org/rfc/rfc2527.txt>
- [RFC2440] J. Callas; L. Donnerhacker; H. Finney und R. Thayer: „OpenPGP Message Format“
<http://www.ietf.org/rfc/rfc2440.txt>
- [IMC] The Internet Mail Consortium: „S/MIME and OpenPGP“
<http://www.imc.org/smime-pgpmime.html>
-

[PKCS] RSA Laboratories: „Public-Key Cryptography Standards (PKCS)“

<http://www.rsasecurity.com/rsalabs/node.asp?id=2124>

[RFC2078] J. Linn: „Generic Security Service Application Program Interface“

<http://www.ietf.org/rfc/rfc2078.txt>

[PKILAW] Charles R. Merrill: „PKI LAW“

<http://www.pkilaw.com>

[LAWSRV] Bert-Jaap Koops: „Crypto Law Survey“

[BSILIT] Bundesamt für Sicherheit und Informationstechnik: „Literatur zum Thema „Digitale Signatur““

<http://www.bsi.de/esig/lit/index.htm>

[SMIME] RSA Security: „S/MIME Central“
