Vorlesung am 12.05.2014

7 Vertrauensmodelle

Problem: Zuordnung eines Schlüssels zum Schlüsselinhaber

- Beispiel 1: Verschlüsselung mit pk, Entschlüsselung mit sk:
  - Bei falscher Zuordnung: Verlust der Vertraulichkeit
- Beispiel 2: Signaturerzeugung mit sk Verifikation mit pk:
  - Bei falscher Zuordnung: Verlust der Datenauthentizität

Wir unterschieden drei Vertrauensmodelle

- Direct Trust: Nutzer erhält pk direkt vom Schlüsselinhaber
- Web of Trust: Nutzer signieren gegenseitig ihre pks
- Hierarchical Trust: pk wird von einer zentralen Instanz verwaltet

Direct Trust Nur für kleine Gruppen anwendbar (z.B. VPNs)
Paarweiser Schlüsselaustausch notwendig:

- 2 Personen: ein Austausch, 3 Personen: 3 mal Austausch
- n Personen: $n(n-1)/2$
- Kommt neuer Partner hinzu: Austausch mit n Personen

Web of Trust Nutzer signieren öffentliche Schlüssel andere Nutzer (und garantieren so für die Authentizität des Schlüssels).
Beispiel (Alice, Bob und Carl):

- Alice signiert Bobs pk, Bob signiert Carls pk

- Alice vertraut Carls pk (prüft mit Bobs pk Carls pk)

Nutzer haben Schlüsselbund (key ring) mit pk anderer Nutzer
Jedem öffentlichen Schlüssel sind zugeordnet

- Name des Schlüsselinhabers
- Owner Trust (5 Stufen, Prüffähigkeit des Schlüsselinhabers)
- Key Legitimacy (3 Stufen, Vertrauen in den öffentlichen Schlüssel) abgeleitet aus Anzahl Signaturen und den zugehörigen Owner Trusts
- Signaturen des öffentlichen Schlüssels

Vorgehen in RFC 2440 (OpenPGP Message Format) beschrieben

Owner Trust

- Grad der Prüftiefe des Schlüsselinhaber für öff. Schlüssel andere Nutzer
  - unknown: keine Informationen verfügbar
  - not trusted: Schlüsselinhaber wird nicht vertraut
  - marginal: Schlüsselinhaber wird teilweise vertraut
  - complete: Schlüsselinhaber wird voll vertraut
  - ultimate: üblicherweise nur für die eigene Person

Key Legitimacy: Grad des Vertrauens in einen öffentlichen Schlüssel

- x = Anzahl Signaturen von Personen mit Owner Trust marginal
- X = erforderliche Anzahl von Signaturen mit Owner Trust marginal (um Schlüssel als authentisch einstufen zu können)
- y = Anzahl Signaturen von Personen, deren Owner Trust complete ist
- Y = Anzahl von Signaturen mit Owner Trust complete (um Schlüssel als authentisch einstufen zu können)
- Vertrauenslevel $L := x/X + y/Y$. Man unterscheidet drei Level:

  - $\underbrace{L = 0}_{\textit{nicht autentisch}}$, $\underbrace{0 < L < 1}_{\textit{teilweise autentisch}}$, $\underbrace{L \geq 1}_{\textit{autentisch}}$

- Übliche Werte $X = 2$ und $Y = 1$, d.h. pk ist authentisch, wenn
  - pk von einer vertrauenswürdigen Person, oder
  - von zwei halb vertrauenswürdigen Personen signiert wurde

Nachteile:

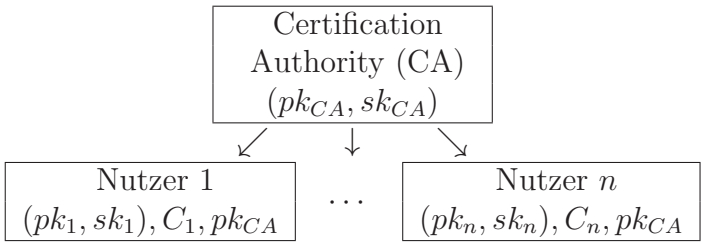
- Einstufung (Owner Trust) erfordert hohes Wissen der Nutzer
- Die Signaturen sind juristisch nicht bindend
- Zurückziehen von Zertifikaten nicht einfach umsetzbar

Vorteile:

- Gegenüber Direct Trust eine deutliche Verbesserung
- Umgesetzt in PGP (Pretty Good Privacy) und GnuPG (Open Source)
- Zahlreiche Schlüsselserver vorhanden
- c't bietet Certification Service für PGP- und GPG-Schlüssel an

Hierarchical Trust Public Key Infrastrukturen

- Öffentliche Schlüssel werden von vertrauenswürdiger Instanz verwaltet
- Nutzer erhalten Zertifikate C_i für ihren öffentlichen Schlüssel pk_i
- Vertrauensanker bildet der öffentliche Schlüssel pk_{CA} der CA



Form/Inhalt eines Zertifikates

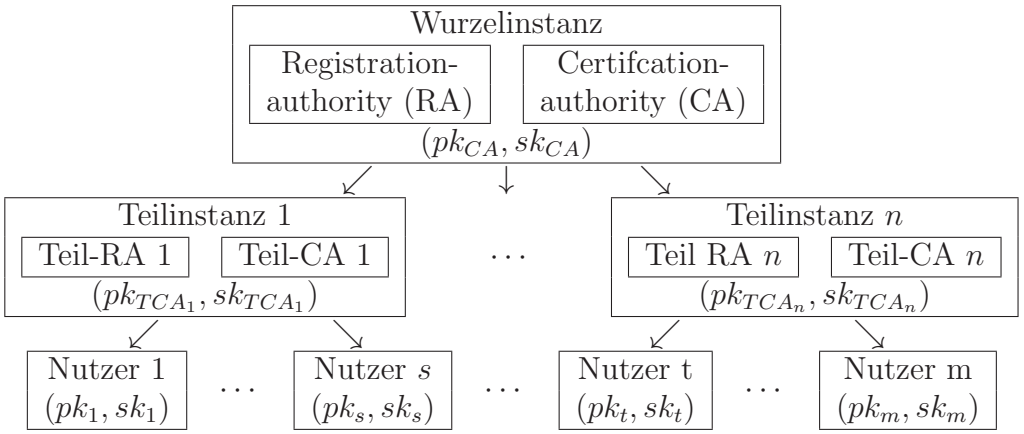
C_i = (Inhalt, Signatur) =

Angaben zum Nutzer: Name, Organisation, ...
öffentlicher Schlüssel pk_i
Verwendeter Algorithmus: z.B. RSA, DSA
Gültigkeitszeitraum
Angaben zur CA: Name, Kontaktdaten, ...
Schlüsselnutzung: Signatur, Verschlüsselung, ...
Signatur durch CA (mit sk_{CA})

Beispiel. Nutzer 1 signiert Dokument m, Nutzer 2 prüft Signatur

Nutzer 1 $(pk_1, sk_1), C_1$	Schlüssel:	Nutzer 2 Schlüssel: pk_{CA}
compute $s := \text{sig}(sk_1, m)$	$\xrightarrow{s, C_1, m}$	verify$(pk_{CA}, \underbrace{\text{inhalt}, \text{signatur}}_{=C_1})$ verify(pk_1, m, s)

Häufig weitere Aufteilung:



Registration Authority

- Legt Sicherheitsrichtlinien fest
- Prüft Zertifizierungsanträge
- Legt Inhalte der Zertifikate fest
- Leitet Anträge an Certification Authority weiter

Certification Authority:

- Stellt die eigentlichen Zertifikate aus

Zurückziehen von Zertifikaten:

- Sicherheitsvorfälle: Schlüsselkompromittierung, Alg. werden unsicher
- Eine Instanz hält sich nicht an Sicherheitsvorgaben
- Hierzu: Führen von Certificate Revokation Lists (CRLs)
- Zusätzlich: Prüfung, ob Zertifikate in CRL enthalten ist

Sicherheitsvorgaben für eine PKI werden in zwei Dokumenten festgelegt:

- Certificate Policy (CP): welche Sicherheitsvorgaben werden eingehalten
- Certificate Practise Statement (CPS): wie werden diese umgesetzt
- RFC 3647:beschreibt wird Aufbau und Inhalt beider Dokumente Internet X.509 PKI CP and Certificate Practise Framework

Es existieren zwei Standards für Zertifikate:

- X509: Eingesetzt zur sicheren Kommunikation im Internet (https)
- Card Verifiable Certificates (cvc): Eingesetzt zur sicheren Kommunikation zwischen/zu Smartcards

X509-Zertifikate: Kodiert nach ASN.1 (Abstract Syntax Notation Nr. 1)

```
Certificate::= SEQUENCE {
   tbsCertificate       TBSCertificate,      TBS: To Be Signed
   signatureAlgorithm   AlgorithmIdentifier, Signaturalgorithmus
   signatureValue       BITSTRING }          Signatur über TBSCert.

TBSCertificate::= SEQUENCE {
   version                    EXPLICIT Version DEFAULT v1,
   serialNumber               CertificateSerialNumber,
   signature                  AlgorithmIdentifier,
   issuer                     Name,
   validity                   Validity,
   subject                    Name,
   subjectPublicKeyInfo       SubjectPublicKeyInfo,
   extensions                 EXPLICIT Extensions OPTIONAL
                              --If present, version MUST be v3 }
```

Erweiterungen (extensions) gibt es erst ab Version 3