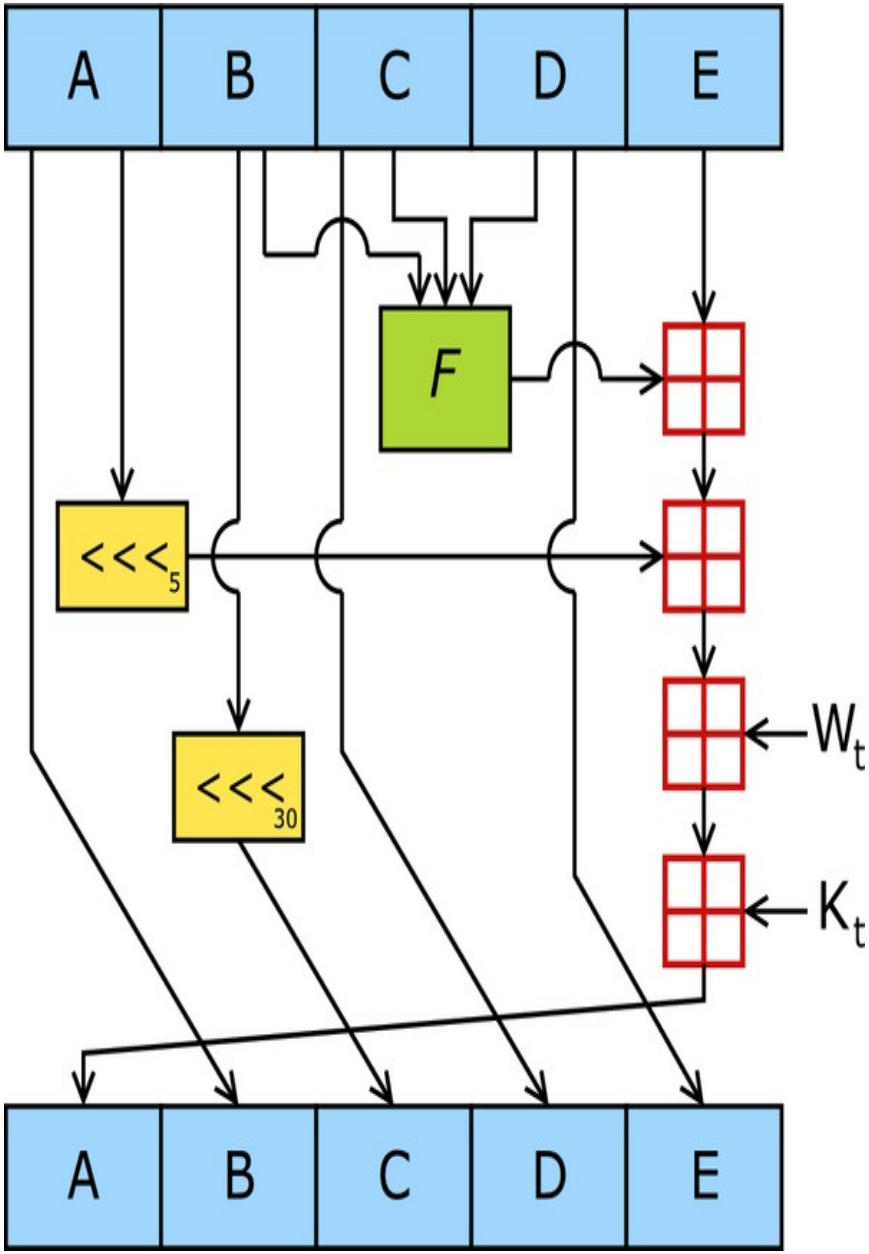
Anforderungen an elektronische Signaturen

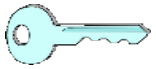
Michel Messerschmidt

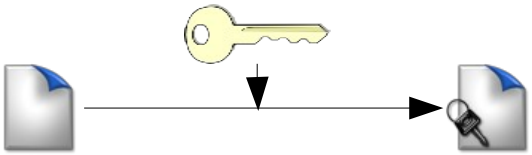
Übersicht

- **Kryptographische Grundlagen**
- **Rechtliche Grundlagen**
- **Praxis**

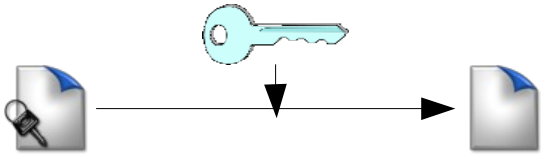


- Verschlüsselung
- Elektronische Signatur
- Zertifikat
- Certification Authority (CA)
- Überprüfung einer Signatur

- **Asymmetrische Verschlüsselung (z. B. RSA, ElGamal)**
  - öffentlicher Schlüssel (public key) Kp 
  - geheimer Schlüssel (secret key) Ks 
  - Verschlüsselung: C = Kp * P

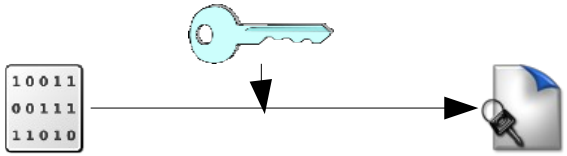


  - Entschlüsselung: P = Ks * C

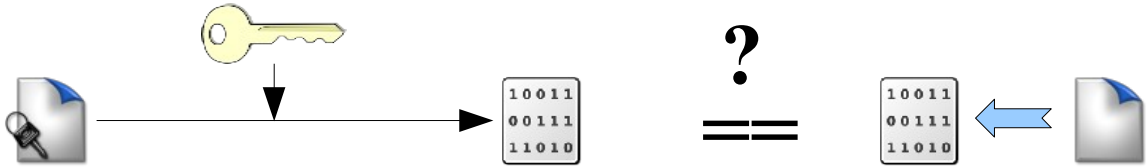


- **garantiert die Vertraulichkeit eines Dokuments**

- **Elektronische Signatur**
  - Dokument P 
  - Hashwert des Dokuments: H(P) 
  - Signierung: S = Ks * H(P)

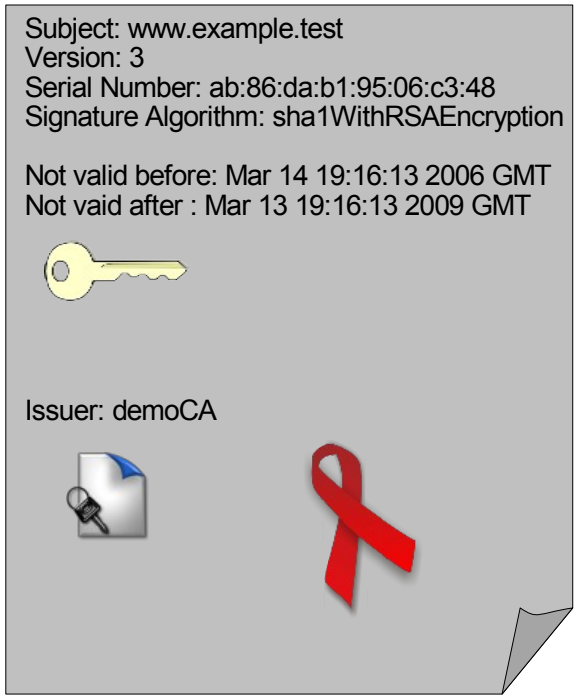


  - Signaturprüfung: H(P) == Kp * S ?

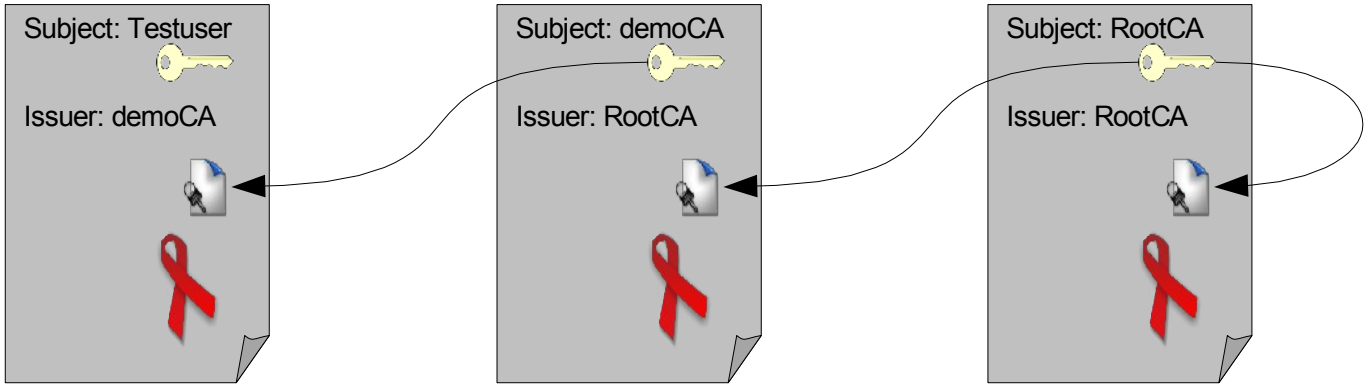


- **garantiert die Integrität eines Dokuments**

- **Was fehlt noch:**
  - Vertrauen in die Signatur
    (Wer hat das Dokument signiert? Wann?)
  - Verteilung des öffentlichen Schlüssels Kp
- **Ein X.509 Zertifikat enthält den öffentlichen Schlüssel Kp und bindet daran weitere Angaben, u.a.:**
  - Identifizierung des Inhabers
  - Gültigkeitszeitraum
  - Beglaubigung des Ausstellers
    (in Form einer elektronischen Signatur)

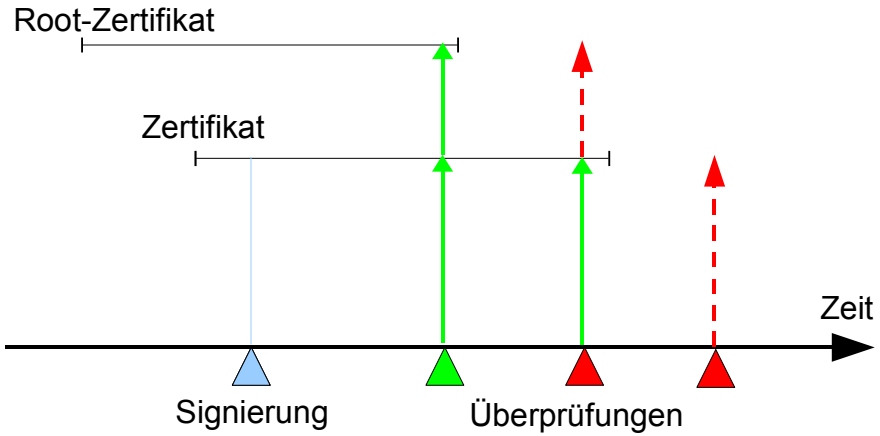


- **Auch Zertifikate kann man fälschen**
  - Vertrauen in das Zertifikat wird durch die Signatur des Zertifikat-Ausstellers (der CA) erreicht
  - Das Zertifikat der CA ist wiederum von dessen Aussteller signiert
  - Es entsteht eine **Zertifikat-Hierarchie**
  - Das **Root-Zertifikat** ist nur selbst-signiert
- **Der Root-CA muss man vertrauen**

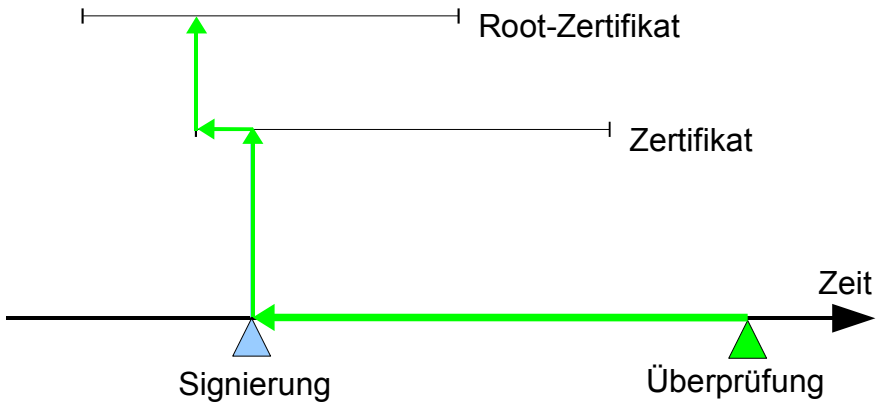


- **Prüfung der Integrität des Dokuments**
  - Stimmt der Hashwert des vorliegenden Dokuments mit dem Hashwert in der Signatur überein?
- **Prüfung der Gültigkeit des Zertifikats**
  - Wurde das Zertifikat widerrufen ?
  - Das Vertrauen in das Dokument entspricht dem Vertrauen in das Zertifikat und der Zertifikathierarchie
- **Es gibt verschiedene Methoden, die Gültigkeit des Zertifikats zu beurteilen:**
  - Schalenmodell
  - Kettenmodell

- **Schalenmodell**
  - Gültigkeit einfach ermittelbar
  - Root-Zertifikate müssen lange gültig sein
    - problematisch wenn Algorithmen unsicher werden
  - weit verbreitet (unter anderem in Windows implementiert)



- **Kettenmodell**
  - Gültigkeitsprüfung relativ aufwändig
  - Gut für Langzeit-Signaturen geeignet
  - Nachsignieren vermeidet Probleme mit Algorithmen
  - bisher kaum verbreitete Implementationen



- **Signaturgesetz**
- **Signatur-Anwendungen**

- **Einfache Signatur**
  - zusätzliche Daten, „die zur Authentifizierung dienen“
    - z. B. eine gescannte Unterschrift

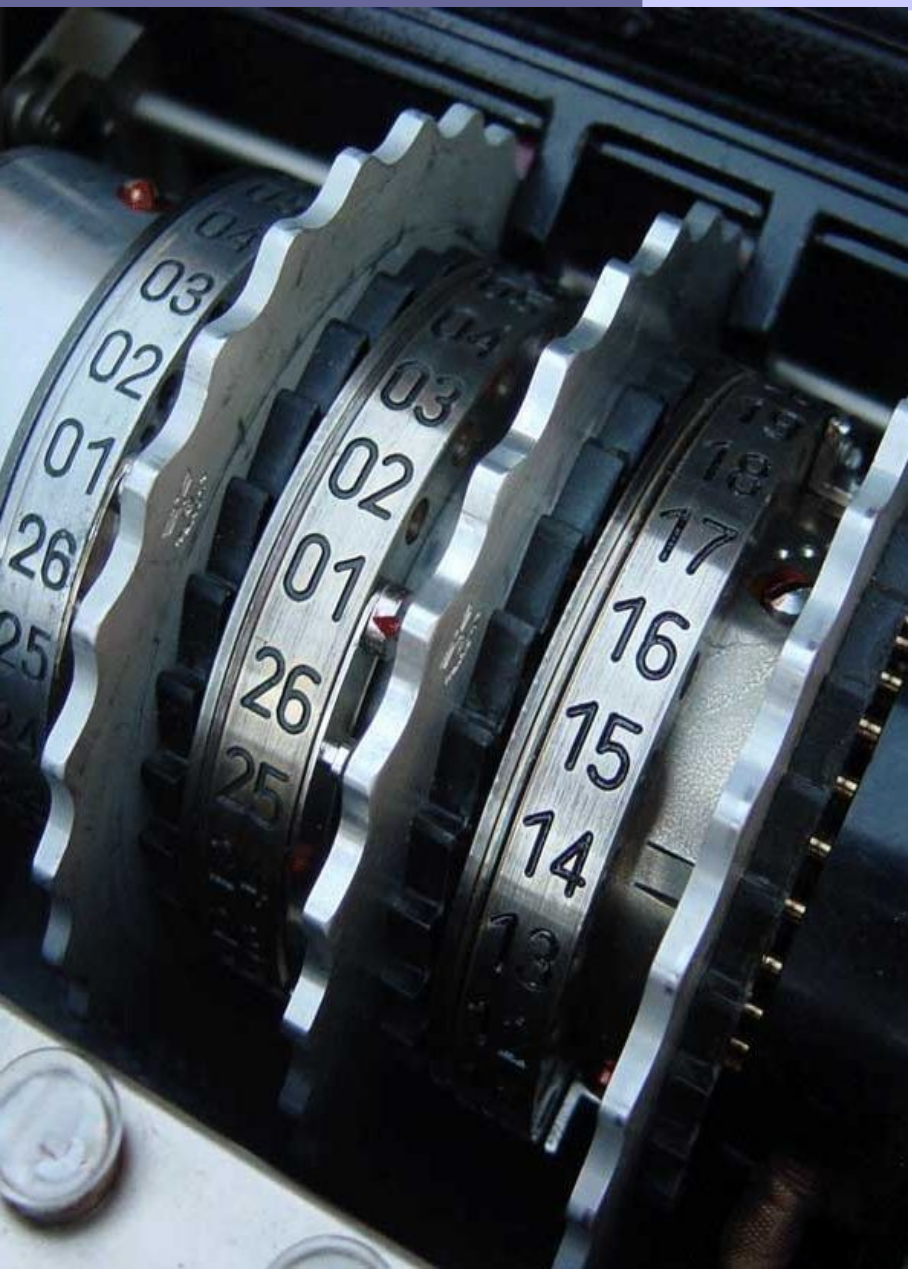
- **Fortgeschrittene Signatur**
  - „ausschließlich dem Signaturschlüssel-Inhaber zugeordnet“
  - ermöglicht Identifizierung des Inhabers
  - kann unter alleiniger Kontrolle des Inhabers erzeugt werden
  - stellt Integrität der Daten sicher
    - an eine natürliche Person gebundene elektronische Signatur

- **Qualifizierte Signatur**
  - alle Anforderungen der fortgeschrittenen Signatur
  - beruht auf einem zum Erzeugungszeitpunkt gültigen qualifizierten Zertifikat
  - „mit einer sicheren Signaturerstellungseinheit erzeugt“
    - mit zusätzlichen Sicherheitsanforderungen
    - Garantie der Authentizität des Inhabers

- **Qualifizierte Signatur mit Anbieter-Akkreditierung**
  - der Anbieter ist zusätzlich bei der BundesNetzAgentur als CA akkreditiert (nach §15 SigG)
    - Garantie der langfristigen Verfügbarkeit

- **Sichere Zeitstempel**
  - erfordert vertrauenswürdige Zeitquelle
- **Elektronische Archivierung**
  - erfordert Integrität der Daten
  - erfordert sichere Zeitstempel
- **Elektronische Rechnungen**
  - erfordert zusätzlich Authentizität des Ausstellers
- **Eigenhändige Unterschrift / Verträge**
  - erfordert zusätzlich Identität des Ausstellers
    - nur mit qualifizierter Signatur möglich

- **Für die rechtliche Gültigkeit ist immer eine Prüfung der Signatur erforderlich**
  - sollte aufgrund des begrenzten Gültigkeitszeitraums möglich schnell erfolgen
  - muss nachweisbar sein
    - Prüfprotokoll muss existieren und mit einem Zeitstempel abgesichert sein



- **Geeignete Algorithmen**
- **Java SDK**
- **Openssl**

- **Die Bundesnetzagentur veröffentlicht jedes Jahr Vorgaben für Algorithmen, die für die nächsten 6 Jahre geeignet sind**
- **Aktuelle Vorgaben**
  - **Geeignete Hashfunktionen**: SHA-224, SHA-256, SHA-384, SHA-512
    - mit Vorbehalt auch noch: RIPEMD-160 (bis 2010), SHA-1 (bis 2009)
    - MD5 ist inzwischen unzureichend
  - **Geeignete Signaturalgorithmen**: RSA, DSA, EC-DSA, ...
    - Empfohlene Schlüssellänge für RSA und DSA: 2048 bit
    - Minimale Schlüssellängen:
      - 1024 bit nur noch bis 2007
      - 1280 bit bis 2008
      - 1536 bit bis 2009
      - 1728 bit RSA bzw. 2048 bit DSA bis 2010
      - 1976 RSA bis 2011
  - Anforderungen an **Zufallsgeneratoren**

- **„keytool“, „jarsigner“, „policytool“**
- **Mit „keytool“ können Signaturschlüssel, Zertifikat-Requests und auch selbst-signierte Zertifikate erstellt werden**
  - Default-Werte sind für heutige Anforderungen nicht geeignet
  - Generierung von DSA-Schlüsseln auf max. 1024 bit beschränkt
  - Besser (Java 1.5.0):

```
keytool -genkey -alias mykey -validity 1095 -keyalg RSA -keysize 2048
-sigalg SHA1withRSA -dname "CN=Michel Messerschmidt,
OU=MeineAbteilung, O=MeineOrganisation, ST=Hamburg, C=DE"
```

  - Oder Schlüssel und Zertifkate sollten auf andere Weise erzeugt und dann mit „keytool“ importiert werden

- **Erzeugen eines Zertifikat-Requests**

```
openssl req -newkey rsa:2048 -days 1095 -out crtreq.pem -keyout
seckey.pem -nodes
```

  - Die Datei „crtreq.pem“ muss an eine CA gesendet werden
  - Diese signiert den Request und sendet das vollständige Zertifikat zurück

- **Erzeugen eines selbst-signierten Zertifikats**

```
openssl req -x509 -newkey rsa:2048 -days 1095 -out self_crt.pem -keyout self_seckey.pem
-nodes
Generating a 2048 bit RSA private key
..............+++
..............+++
writing new private key to 'self_seckey.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:DE
State or Province Name (full name) [Some-State]:Hamburg
Locality Name (eg, city) []:Hamburg
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Michel Messerschmidt
Organizational Unit Name (eg, section) []:Web server
Common Name (eg, YOUR name) []:testserver.testnet
Email Address []:webmaster@testserver.testnet
```

Links

- **Bundesnetzagentur (ehemals RegTP, stellt Gesetzestexte sowie Listen zertifizierter Anbieter, Hard- und Software bereit):**
  http://www.bundesnetzagentur.de/
- **Verständliche Informationen zu Kryptographie und elektronischen Signaturen:**
  http://www.cryptoshop.com/de/knowledgebase/
- **Alles über den richtigen Umgang mit Sicherheitsmechanismen (von Peter Gutmann):**
  http://www.cs.auckland.ac.nz/~pgut001/
- **Ausführliche Informationen zu Public Key Infrastructure (von Stefan Kelm):**
  http://www.pki-page.org/
- **Ausführliche Informationen zu Kryptographie (von Bruce Schneier):**
  http://www.schneier.com/resources.html
- **Elektronische Rechnungen gesetzeskonform prüfen:**
  http://www.elektronische-steuerpruefung.de/loesung/kirmes_erech.htm
- **Öffentliche Initiativen zur Förderung elektronischer Signaturen:**
  http://www.izn.niedersachsen.de/master/C6023924_N5557488_L20_D0_I3654280.html,
  http://www.signaturbuendnis.de,
  http://www.archisig.de/