Mathematische Grundlagen asymmetrischer Systeme

- Modulo-Rechnung
  - Grundlagen
  - Erweiterter Euklidscher Algorithmus
- Systeme auf der Basis des diskreten Logarithmus
  - primitive Wurzel
  - diskreter Logarithmus Problem
- Systeme auf der Basis der Faktorisierungsannahme
  - Faktorisierungsannahme
  - Primzahlzerlegung

Erweiterter Euklidscher Algorithmus

- Anwendung
  - Bestimmung von ggT(a,b) und Ermittlung des multiplikativen Inversen von b im Restklassenring modulo a, d. h. zur Berechnung von b^{-1} aus der Beziehung $b \cdot b^{-1} = 1 \bmod a$
- Algorithmus
  - Seien $a, b \in \boldsymbol{N}+1,\ a > b$
  - Setze $r_{-2} = a$ $\quad s_{-2} = 0$ $\quad t_{-2} = 1$
    $r_{-1} = b$ $\quad s_{-1} = 1$ $\quad t_{-1} = 0$
  - Berechne c_k, r_k, s_k und t_{-k} nach folgenden Beziehungen:

    $c_k = r_{k-2} \text{ DIV } r_{k-1}$

    $r_k = r_{k-2} \text{ MOD } r_{k-1}$

    $s_k = c_k s_{k-1} + s_{k-2}$

    $t_k = c_k t_{k-1} + t_{k-2}$
  - Abbruchbedingung : $r_k = 0$
  - Es gilt: $b \cdot s_{k-1} - a \cdot t_{k-1} = (-1)^k \cdot r_{k-1}$
  - Ergebnisse: $r_{k-1} = \text{ggT}(a,b)$

    $s_{k-1} \cdot b = (-1)^k \bmod a$, falls $\text{ggT}(a,b) = 1$

$c_k = r_{k-2}$ DIV r_{k-1} $\quad r_k = r_{k-2}$ MOD r_{k-1} $\quad s_k = c_k s_{k-1} + s_{k-2}$ $\quad t_k = c_k t_{k-1} + t_{k-2}$

- Beispiel 1
  - Gegeben: a=10 b=4
  - Gesucht: ggt(a,b)

k	c_k	r_k	s_k	t_k
-2		10 = a	0	1
-1		4 = b	1	0
0	2	2 = ggt	2	1
1	2	0 (Abbruch)		

$c_k = r_{k-2} \text{ DIV } r_{k-1}$ $r_k = r_{k-2} \text{ MOD } r_{k-1}$ $s_k = c_k s_{k-1} + s_{k-2}$ $t_k = c_k t_{k-1} + t_{k-2}$

- Beispiel 2
  - Gegeben:
    p=53 q=61 n=p·q=3233 Φ(n)=(p-1)·(q-1)=3120 c=523
  - Gesucht:
    $c \cdot c^{-1} = 1 \bmod \Phi(n)$, d.h. Multiplikatives Inverses zu c mod Φ(n)

k	c_k	r_k	s_k	t_k
-2		3120 = a = Φ	0	1
-1		523 = b = c	1	0
0	5	505	5	1
1	1	18	6	1
2	28	1 = ggt	173	29
3	18	0 (Abbruch)		

- Beispiel 2 (Forts.)

  – Es gilt:

$s_{k-1} \cdot b$	$=$	$(-1)^k$	mod a
$173 \cdot 523 =$	$(-1)^3$	mod 3120	
90479	$=$	-1	mod 3120
3119	$=$	-1	mod 3120
-1	$=$	-1	mod 3120

  – Da $(-1)^k = (-1)^3 = -1$, muss noch mit -1 multipliziert werden.

$-s_{k-1} \cdot b$	$=$	$-(-1)^k$	mod a
c	$=$	$-s_{k-1} = -173 = -173 + a$	
c	$=$	2947	

Eulersche-Φ-Funktion

- Def. Eulersche Φ-Funktion
  - Für eine beliebige ganze Zahl n bildet die Menge $\mathbf{Z}_n^*$ der ganzen Zahlen modulo n, die zu n teilerfremd sind, eine multiplikative Gruppe. Die Ordnung dieser Gruppe ist $\Phi(n)$.
    - Beispiel: $\Phi(9) = 6$ $\mathbf{Z}_9^* = \{1,2,4,5,7,8\}$
  - Für den Sonderfall $n = p \in \boldsymbol{P}$ gilt $\Phi(p) = p-1$.
- Satz von Euler
  - Für ein beliebiges x mit $(1 \leq x < n)$, das zu n teilerfremd ist bzw. $x \in \mathbf{Z}_n^*$, gilt $x^{\Phi(n)} = 1 \bmod n$.
- Euler-Fermat-Identität (kleiner Satz von Fermat)
  - Mit $\Phi(p) = p-1$ ($p \in \boldsymbol{P}$) folgt aus dem Satz von Euler:

    $x^{p-1} = 1 \bmod p \quad (1 \leq x \leq p-1)$.
- Wenn n das Produkt zweier Primzahlen $n = p \cdot q$ ist, gilt

  $x^{\Phi(p \cdot q)} = x^{(p-1)\cdot(q-1)} \bmod p \cdot q$.

Primitive Wurzel

- Definition
  - Eine beliebige ganze Zahl a im Bereich $1 \leq a < n$ heißt primitive Wurzel von n, wenn gilt

    $ggT(a,n) = 1$ *und*

    $a^d \neq 1 \bmod n$ für alle d mit der Bedingung $1 \leq d < \Phi(n)$
- Theorem
  - Die ganze Zahl n hat genau dann eine primitive Wurzel, wenn $n = 1, 2$ oder 4 ist oder die Form p^k oder $2p^k$ hat, wobei p eine ungerade Primzahl ist.
  - Wenn n eine primitive Wurzel hat, dann hat n genau $\Phi(\Phi(n))$ primitive Wurzeln.
- Vermutung
  - Jede ganze Zahl, die keine Quadratzahl ist, ist die primitive Wurzel einer Primzahl.

Diskreter Logarithmus

- **Definition diskreter Logarithmus**
  - Sei p eine beliebige ganze Zahl, die eine primitive Wurzel a hat.
    Wenn für ein beliebiges c mit $0 \leq c < \Phi(p)$

    $$b = a^c \mod p$$

    gilt, dann ist c der diskrete Logarithmus zur Basis a modulo p oder auch

    $$c = \log_a b \mod p.$$
- **Problemstellung für den Angreifer**
  - Öffentlich bekannt sind a, b, p.
  - Geheim ist c. Erfährt ein Angreifer c, ist das System gebrochen.
  - Folglich möchte ein Angreifer c ermitteln.
- **Beispiel**
  - $p = 3137$ und $a = 577$ öffentlich; $c = 1374$ geheim
  - $b = a^c \mod p = 858$ öffentlich
  - $c = \log_a b \mod p = \log_{577} 858 \mod 3137 =$ **?** (Angreifersicht)

- Algorithmen zur Berechnung des diskreten Logarithmus
  - Baby-Step, Giant-Step, Index-Calculus-Alg., Adleman-Alg.
  - Laufzeit zur Berechnung des diskreter Log. mod p mit p $\in$ ***P***

$$e^{(1+O(1))(\log p \cdot \log(\log p))^{1/2}}$$

  - Rechenzeiten bei 10^8 Operationen pro sek. für verschiedene Größenordnungen von p:

p	Anzahl Ops.	benötigte Zeit in Jahren
$\approx 10^{100}$	$7{,}9 \cdot 10^{22}$	$2{,}5 \cdot 10^{7}$
$\approx 10^{200}$	$1{,}8 \cdot 10^{34}$	$5{,}7 \cdot 10^{19}$
$\approx 10^{300}$	$1{,}8 \cdot 10^{43}$	$5{,}7 \cdot 10^{28}$
$\approx 10^{400}$	$9{,}5 \cdot 10^{50}$	$4{,}8 \cdot 10^{36}$
$\approx 10^{500}$	$7{,}4 \cdot 10^{57}$	$4{,}8 \cdot 10^{43}$

Vergleich: Logarithmus

$\log_a b$ (a>0; a≠1; b>0) ist diejenige reelle Zahl c, für die gilt $a^c=b$.

c = $\log_a b$ wird z.B. gelöst mit

$$\log_a b = \frac{\log_x b}{\log_x a}$$

- Beispiel 1: Wieviel Bit benötigt man, um die Zahlen zwischen 0 und 255 binär zu kodieren?

$$\log_2 256 = \frac{\ln 256}{\ln 2} = 8\,\text{Bit}$$

- Beispiel 2: Wieviel Bit benötigt man, um die Zahlen bis 10^{200} im Binärcode darzustellen?

$$10^{200} \le 2^x \qquad x \ge \log_2 10^{200} = \frac{\log_{10} 10^{200}}{\log_{10} 2} = \frac{200}{\log_{10} 2} \qquad x \ge 665\,\text{Bit}$$

- Beispiel 3: a = 577; b = 858; c = ?

$$c = \log_a b = \log_{577} 858 = \frac{\ln 858}{\ln 577} = 1{,}0624$$

Faktorisierungsannahme

- Seien p und q zwei große ($|p| \approx |q| \approx 500 \dots 1500$ Bit), unabhängig und zufällig gewählte Primzahlen. p und q werden geheim gehalten. Das Produkt n aus p und q wird veröffentlicht: $n=p \cdot q$.

- Annahme: Für jeden schnellen Faktorisierungsalgorithmus F(n) wird die Wahrscheinlichkeit, dass F(n) eine Zahl $n=p \cdot q$ tatsächlich faktorisieren kann, schnell kleiner, je größer die Länge $|p|$ und $|q|$ der Faktoren ist.

  - Es ist zwar mit vernünftigem Aufwand möglich, Primzahlen p und q zu finden und diese zu multiplizieren. Es ist aber nicht mit vernünftigem Aufwand möglich, die Primfaktoren von n zu finden.
  - Die öffentlich ausführbare Funktion (Verschlüsseln bzw. Signaturtest) kommt mit dem öffentlichen n aus. Die private Funktion (Entschlüsseln bzw. Erzeugen einer Signatur) nutzt p und q.
  - Dass Faktorisierung schwer ist, ist bisher nicht bewiesen.

A will B die Nachricht m schicken.

B:

$p_B \in P$ und a primitive Wurzel von p_B wählen

x_B mit $1 \leq x_B \leq p_B-1$ wählen

$y_B = a^{x_B} \mod p_B$ berechnen

x_B bleibt geheim!

a, p_B und y_B sind auf key server veröffentlicht

A:

liest Eintrag für B: a, p_B und y_B

x_A mit $1 \leq x_A \leq p_B-1$ geheim wählen

$y_A = a^{x_A} \mod p_B$ berechnen

Key Agreement:

$k_{AB} = y_B^{x_A} \mod p_B$ berechnen

Verschlüsselung:

$s = E(k_{AB}, m)$ → s und y_A →

B:

$k_{BA} = y_A^{x_B} \mod p_B$ berechnen

$m = E^{-1}(k_{BA}, s)$ entschlüsseln

A will B die Nachricht m schicken.

B:

$p_B \in P$ und a primitive Wurzel von p_B wählen

x_B mit $1 \leq x_B \leq p_B$-1 wählen

$y_B = a^{x_B}$ mod p_B berechnen

x_B bleibt geheim!

Beispielrechnung mit p_B=23, a=5, x_A=2, x_B=3: 10 5^3 23

a, p_B und y_B sind auf key server veröffentlicht

A:

liest Eintrag für B: a, p_B und y_B

x_A mit $1 \leq x_A \leq p_B$-1 geheim wählen

$y_A = a^{x_A}$ mod p_B 2 = 5^2 mod 23

Key Agreement:

$k_{AB} = y_B^{x_A}$ mod p_B 8 = 10^2 mod 23

Verschlüsselung:

$s = E(k_{AB}, m)$ 8

s und y_A →

B:

$k_{BA} = y_A^{x_B}$ mod p_B 8 = 2^3 mod 23

$m = E^{-1}($ 8 $, s)$ entschlüsseln

Diffie-Hellman-Key-Exchange mit Forward Secrecy

A will B die Nachricht m schicken.

B:

$p_B \in P$ und a primitive Wurzel von p_B wählen

x_B mit $1 \leq x_B \leq p_B$-1 wählen

$y_B = a^{x_B}$ mod p_B berechnen

x_B bleibt geheim!

a, p_B und y_B Wenigstens y_B wird je Sitzung neu erzeugt.

A:

liest Eintrag für B: a, p_B und y_B

x_A mit $1 \leq x_A \leq p_B$-1 geheim wählen

$y_A = a^{x_A}$ mod p_B berechnen

Key Agreement:

$k_{AB} = y_B^{x_A}$ mod p_B berechnen

Verschlüsselung:

$s = E(k_{AB}, m)$ → s und y_A →

Forward Secrecy: k lässt sich nach Beendigung der Sitzung nicht mehr aus dem verwendeten Langzeitschlüssel (a, p_B , y_B) rekonstruieren.

B:

$k_{BA} = y_A^{x_B}$ mod p_B berechnen

$m = E^{-1}(k_{BA}, s)$ entschlüsseln

Diffie-Hellman-Key-Exchange

- Berechnung des Kommunikationsschlüssels

  k_{AB} bzw. k_{BA} erfolgt durch

  $k_{AB} = y_B^{x_A} \bmod p_B$ bei A und

  $k_{BA} = y_A^{x_B} \bmod p_B$ bei B.

- Nachweis

  $$k_{AB} = y_B^{x_A} = (a^{x_B})^{x_A} = (a^{x_A})^{x_B} = y_A^{x_B} = k_{BA} \quad (\bmod p_B)$$

- Angreifer muss zum Brechen x_A oder x_B ermitteln, d.h. er muss berechnen:

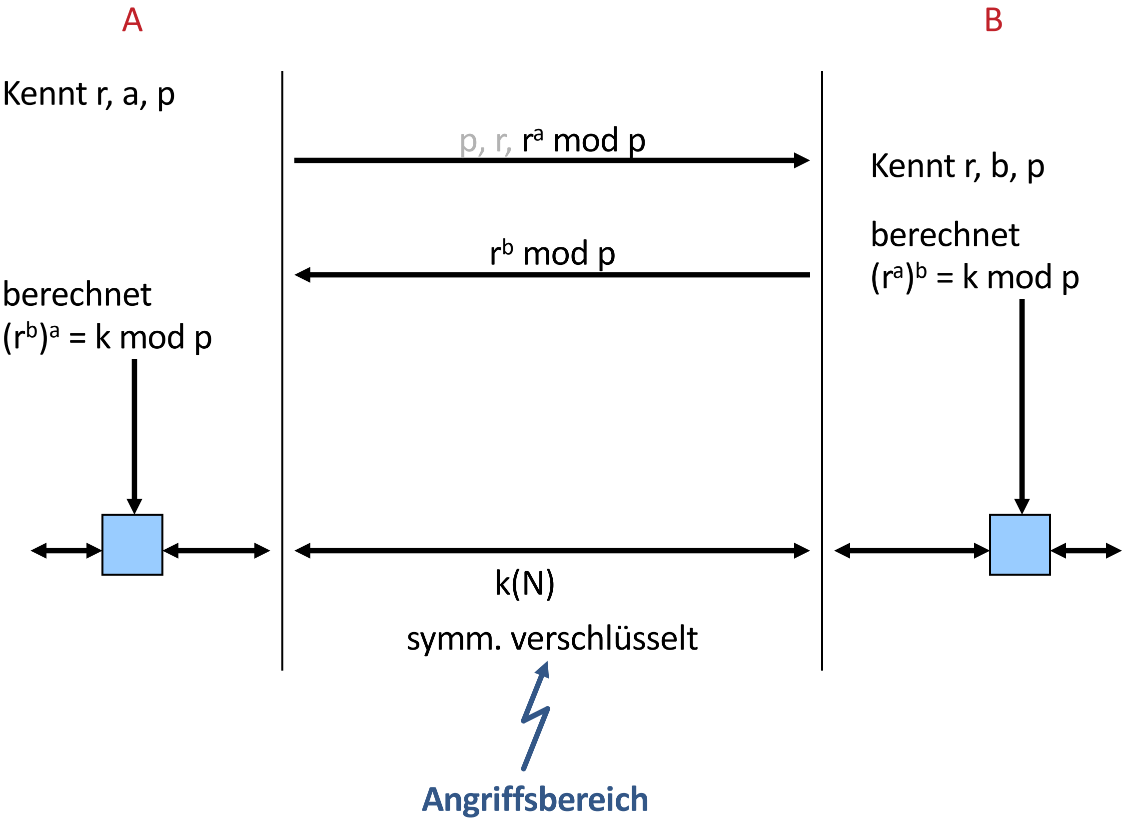
  $x_A = \log_a y_A \quad \bmod p_B$ oder

  $x_B = \log_a y_B \quad \bmod p_B$

- Sicherheit
  - Verfahren ist sicher gegen einen passiven Angreifer.
  - Verfahren ist unsicher gegen einen aktiven Angreifer (Maskerade).

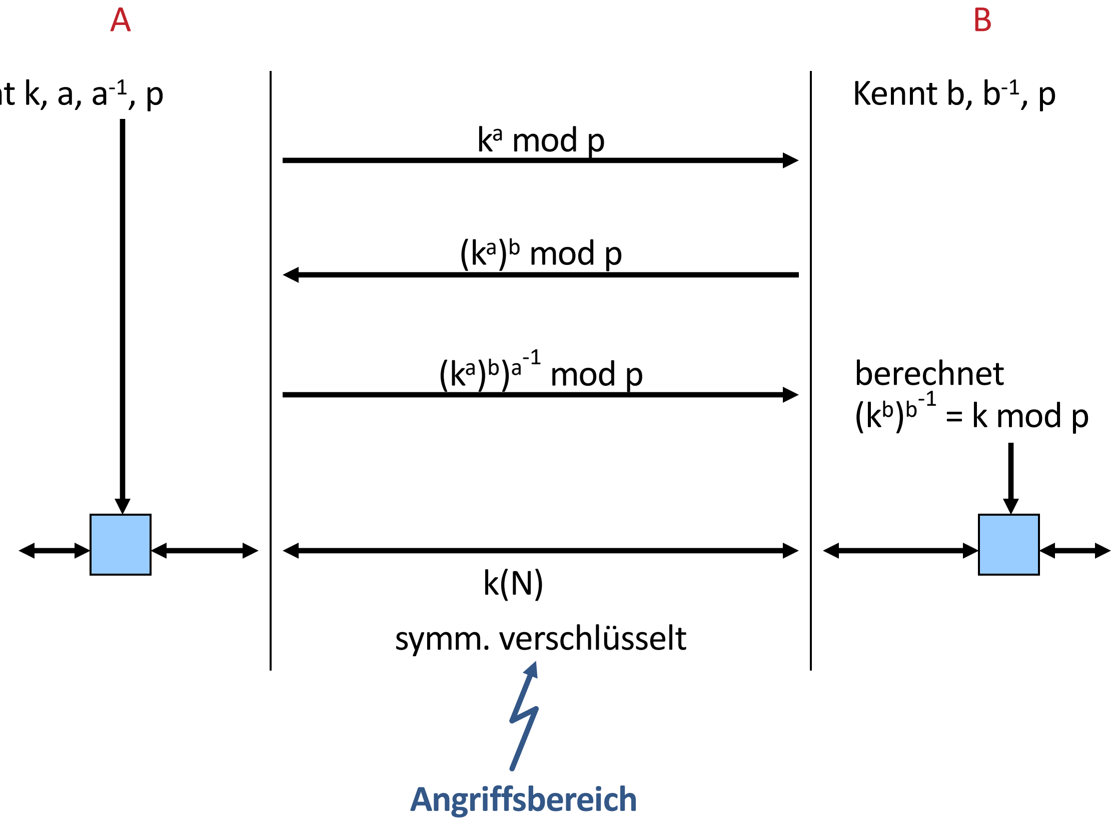
Asymmetrische Schlüsselvereinbarung (v1)

- Wert von k kann nicht festgelegt werden (ist zufällig)



Asymmetrische Schlüsselvereinbarung (v2)

- Teilnehmer A kann k festlegen



Konzelationssystem nach ElGamal

ElGamal basiert auf der Schwierigkeit der Berechnung des diskreten Log

- Schlüsselgenerierung
  - wähle global:
    - · $p \in \boldsymbol{P}$ — öffentlich
    - · a primitive Wurzel von p — öffentlich
  - jeder Tln. wählt:
    - · geheimen Schlüssel k_i ($k_i < p-1$) — geheim
    - · berechnet $-k_i \mod (p-1)$ — geheim
    - · $y_i = a^{-k_i} \mod p$ — öffentlich
- Verschlüsselung
  - A will Nachricht m ($m < p$) an B schicken
  - A besorgt sich p, a, y_B
  - A wählt Zufallszahl z ($z < p$)
  - A berechnet $c = y_B{}^z \cdot m \mod p$
  - A sendet an B: $a^z \mod p, c$
- Entschlüsselung
  - B berechnet $m^* = (a^z)^{k_B} \cdot c \mod p$

$$A \xrightarrow{m} B$$

Konzelationssystem nach ElGamal: Beispiel

- Schlüsselgenerierung
  - Global öffentlich: $p = 3137$ und $a = 577$
  - Teilnehmer B: $k_B = 1762$ geheim

    $-k_B \bmod (p-1) = -1762 \bmod 3136 = -1762 + 3136 = 1374$ geheim

    $y_B = a^{-k_B} \bmod p = 577^{1374} \bmod 3137 = 858$
- Verschlüsselung
  - **A** will **B** vertraulich die Nachricht $m = 2115$ schicken.
  - **A** wählt $z = 932$ geheim.
  - berechnet $a^z \bmod p = 577^{932} \bmod 3137 = 1852$
  - berechnet $y_B^z \bmod p = 858^{932} \bmod 3137 = 749$
  - berechnet $c = y_B^z \cdot m \bmod p = 749 \cdot 2115 \bmod 3137 = 3087$
  - schickt $a^z = 1852$ und $c = 3087$ an **B**
- Entschlüsselung
  - **B** berechnet

    $(a^z)^{k_B} \cdot c \bmod p = 1852^{1762} \cdot 3087 \bmod 3137 = 2115$

Signatursystem nach ElGamal

- Schlüsselgenerierung
  - wähle global:
    - · $p \in \boldsymbol{P}$ — öffentlich
    - · a primitive Wurzel von p — öffentlich
  - jeder Tln. wählt:
    - · $x_i \in \boldsymbol{Z}^*_{p-1}$ — geheim
    - · berechnet $\mathbf{y_i = a^{x_i} \ mod\ p}$ — öffentlich
- Signatur
  - A wählt:
    - · Zufallszahl k mit $k \in \boldsymbol{Z}^*_{p-1}$ bzw. k relativ prim zu p-1, d.h. ggt(k,p-1)=1
  - A berechnet:
    - · k^{-1} mod (p-1) und
    - · $r = a^k \mod p$
    - · bildet Hash-Wert von m mit h(m) < p und löst die Kongruenz
      $h(m) = (x_A \cdot r + k \cdot s) \mod (p-1)$ nach s auf: $s = k^{-1}(h(m) - x_A \cdot r) \mod (p-1)$
  - A bildet sig = (s, r) und sendet m, sig

$$A \xrightarrow{m,\ sig} B$$

- Test
  - B berechnet:
    - · $t_1 = a^{h(m)} \mod p$
    - · $t_2 = y_A{}^r \cdot r^s \mod p$
  - B vergleicht:
    - · $t_1 = t_2 \longrightarrow$ gültige Signatur
    - · $t_1 \neq t_2 \longrightarrow$ ungültige Signatur

- Schlüsselgenerierung
  - wähle unabh. und zufällig $p, q \in \boldsymbol{P}$ mit $|p| \approx |q|$ und $p \neq q$
  - berechne $n = p \cdot q$
  - wähle c mit $3 \leq c < \Phi(n)$ und
    $ggt(c, \Phi(n))=1$ mit $\Phi(n) = (p-1)(q-1)$
  - berechne d mittels p, q, c als multiplikatives Inverses von $c \bmod \Phi(n)$
    $c \cdot d \equiv 1 \bmod \Phi(n)$

	Konzelationssystem	Signatursystem
öffentl.	c, n	d (hier meist t genannt), n
geheim	d, p, q	c (hier meist s genannt), p, q

RSA basiert auf der Faktorisierungsannahme

- Ein Sicherheitsbeweis von RSA ist bisher nicht bekannt.

	Verschlüsselung:	Signatur:
	A will Nachricht m $(1 < m < n)$ an B schicken $A \xrightarrow{c(m)} B$	A will Signatur einer Nachricht m $(1 < m < n)$ von B testen $A \xleftarrow{m,\ sig(m)} B$
	A besorgt sich öffentliche Parameter von B: c bzw. t, sowie n	
naiv:	$c(m) := m^c \bmod n$	$sig_s(m) := m^s \bmod n$
sicher:	$c(m) := (z, m, h(z, m))^c \bmod n$	$sig_s(m) := (h(m))^s \bmod n$

	Entschlüsselung:	Signaturtest:
naiv:	$m^* = (m^c)^d \bmod n$	$m^* = (m^s)^t \bmod n$ $m^* =?\ m' \longrightarrow$ out(ok)
sicher:	$(z^*, m^*, y) = c(m)^d \bmod n$ $y =?\ h(z^*, m^*) \longrightarrow$ out(m)	$h(m)^* = ((h(m))^s)^t \bmod n$ $h(m)^* =?\ h(m') \longrightarrow$ out(ok)

RSA-Verfahren: Angriffe

- Raten von Klartextblöcken
  - Angreifer kann wahrscheinliche Klartextblöcke raten, mit c verschlüsseln und mit abgefangenen Schlüsseltexten vergleichen.

- Verhinderung
  - Zufallszahl in Klartext hineinkodieren

$$\text{deterministisches Kryptosystem} \xrightarrow{\text{Zufallszahl in Klartext hineinkodieren}} \text{indeterministisches Kryptosystem}$$

RSA besitzt multiplikative Struktur

- Passiver Angriff auf naives Signatursystem (Davida)
  - Angenommen, Angreifer kennt zwei Signaturen m_1^s und m_2^s sowie die Nachrichten m_1 und m_2 und kann eine dritte Signatur m_3^s bilden. m_3 ist jedoch nicht beliebig wählbar.

    $m_3^s \quad = m_1^s \cdot m_2^s \bmod n$

    $m_3 \quad = m_1 \cdot m_2 \bmod n$

    gilt, da $m_1^s \cdot m_2^s = (m_1 \cdot m_2)^s$
- Homomorphismus bezüglich Multiplikation

$$\begin{tikzcd}
(m_1,m_2) \arrow[r, "\cdot"] \arrow[d, "x^s"] & m_1 \cdot m_2 \arrow[d, "x^s"] \\
(m_1^s,m_2^s) \arrow[r, "\cdot"] & m_1^s \cdot m_2^s = (m_1 \cdot m_2)^s
\end{tikzcd}$$

RSA-Verfahren: Angriffe

- Aktiver Angriff zum selektiven Brechen von RSA nach Judy Moore
  - Angreifer möchte Schlüsseltextblock s_3 entschlüsselt haben
  - wählt Zufallszahl r mit $1 \leq r < n$
  - berechnet multiplikatives Inverses mod n: r^{-1}
  - berechnet $s_2 := s_3 \cdot r^c \bmod n$
  - lässt s_2 entschlüsseln, d.h. Angreifer erhält s_2^d
  - weiß $s_2^d \equiv (s_3 \cdot r^c)^d \equiv s_3^d \cdot r^{c \cdot d} \equiv s_3^d \cdot r \bmod n$
  - berechnet $s_3^d \equiv s_2^d \cdot r^{-1} \bmod n$

Angriff wird nutzbar gemacht für blinde Signaturen (Chaum 1985)

- Verhinderung der Angriffe (aktiv und passiv)
  - Konzelation: Hinzunahme eines Redundanzprädikates, z.B. einer Zufallszahl, so dass das Multiplizieren zweier Klartextblöcke keinen dritten Klartextblock mit passender Redundanz ergibt; alternativ: vor Verschlüsselung Hashwert an Nachricht anhängen
  - Signatur: Signatur des Hashwertes h(m) der Nachricht m. h ist eine kollisionsfreie Hashfunktion. Das Finden einer Kollision, d.h. $h(m) = h(m^*)$ mit $m \neq m^*$ ist ein schwieriges Problem.

Blinde Signatur mit RSA-Verfahren

- Teilnehmer möchte Nachricht m signiert haben, ohne dass der Signierer die Nachricht m selbst zur Kenntnis bekommt
  - wählt Zufallszahl r mit $1 \leq r < n$
  - berechnet multiplikatives Inverses mod n: r^{-1}
  - Blendet die Nachricht m, d.h. berechnet

    $$w := m \cdot r^t \mod n$$

  - lässt w signieren, d.h. erhält w^s

  - Er weiß

    $$w^s = (m \cdot r^t)^s = m^s \cdot r^{t \cdot s} = m^s \cdot r \pmod n$$

  - Entblendet die Nachricht, d.h. berechnet

    $$sig(m) = m^s \cdot r \cdot r^{-1} \pmod n$$

- Anwendungsbeispiel: Anonyme digitale Zahlungssysteme

Blinde Signatur mit RSA-Verfahren

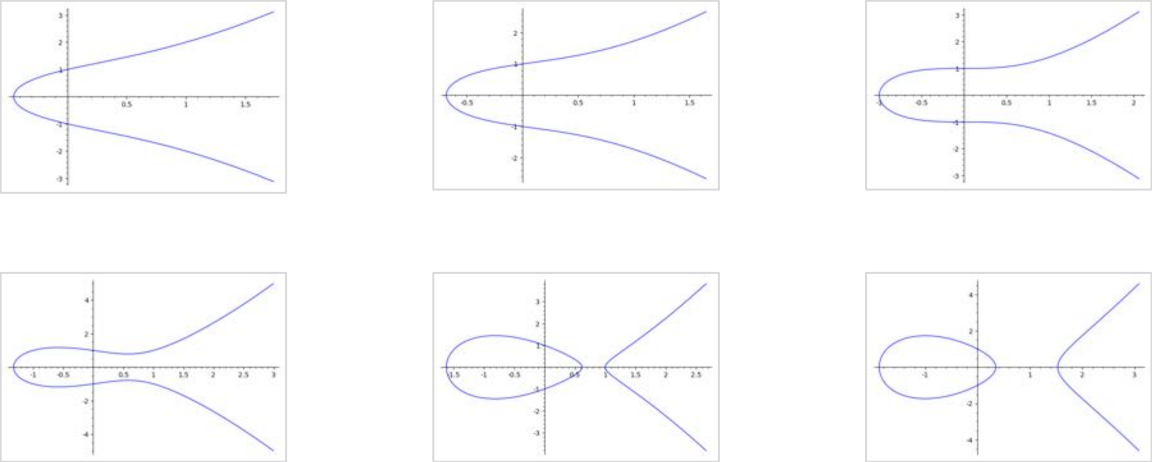
- Anwendungsbeispiel: Anonyme digitale Zahlungssysteme
- Bank erfährt nichts über Zahlungsflüsse ähnlich Bargeld
- Signierer = Bank

- Geld abheben:
  - Kunde schickt geblendete digitale Banknote w an Bank
  - Bank belastet Konto des Kunden mit Gegenwert
  - Bank signiert w und schickt w^s zurück an Kunden
  - Kunde entblendet Banknote und erhält sig(m)
- Bezahlen:
  - Kunde kauft bei Händler ein und bezahlt mit Banknote sig(m)
  - Händler löst sig(m) bei Bank ein
  - Bank prüft sig(m) auf Gültigkeit (korrekte Signatur und nicht bereits eingelöst)
  - Bank schreibt Händler Gegenwert auf seinem Konto gut

Elliptic Curve Cryptography (ECC)

- Elliptische Kurve
  - Eine elliptische Kurve $\boldsymbol{E}$ über den reellen Zahlen $\mathbb{R}$ ist die Menge aller Punkte (x, y), die die Gleichung $y^2 = x^3 + ax + b$ erfüllen ($a, b \in \mathbb{R}$ und $-4a^3 - 27b^2 \neq 0$), zusammen mit dem sogenannten Punkt im Unendlichen ∞.

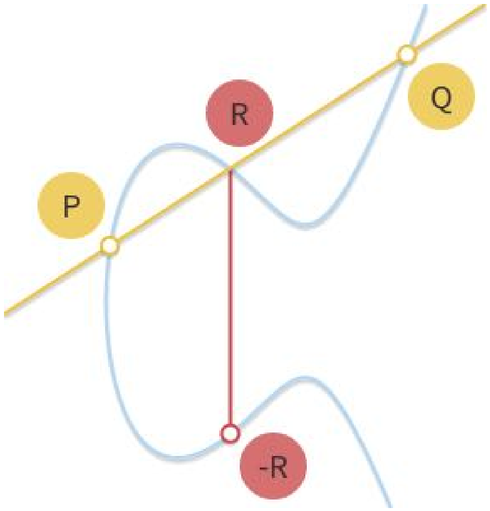
- Beispiel: Verschiedene Kurven mit $a \in \{-3, \ldots, 2\}, b = 1$:



Elliptic Curve Cryptography (ECC)

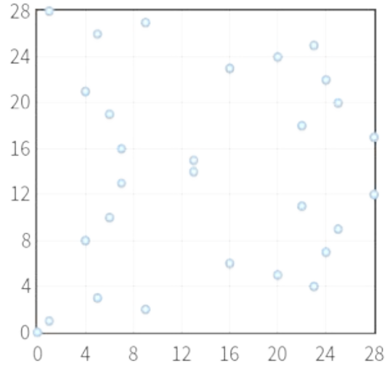
- Zusammen mit der Punktaddition bilden die Punkte einer elliptischen Kurve eine algebraische (kommutative) Gruppe über den Punkten $P_i = (x_i, y_i)$
  - Abgeschlossenheit: $P_1 + P_2 = P_3 \in E$
  - Assoziatiität: $(P_1 + P_2) + P_3 = P_1+(P_2+P_3)$
  - Neutrales Element: $P_i + \infty = P_i$
  - Inverses Element $-P_i$: $P_i + (-P_i) = \infty$
  - Kommutativität: $P_1 + P_2 = P_2 + P_1$

Geometrische Interpretation der Punktaddition: P + Q = -R

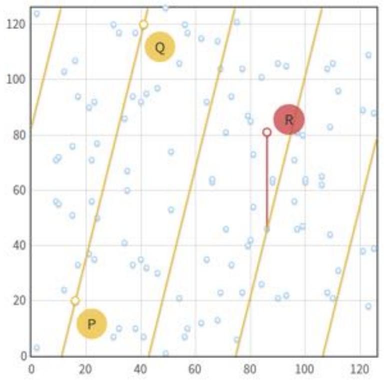


- Skalare Multiplikation wird als mehrfache Addition eines Punktes mit sich selbst betrachtet: $nP = P + \cdots + P$

- Elliptische Kurve über $\mathbb{Z}_p$
  - Eine Elliptische Kurve $\boldsymbol{E}$ über $\mathbb{Z}_p$ ist die Menge aller Punkte $(x, y) \in \mathbb{Z}_p{}^2$, die die Gleichung $y^2 \equiv x^3 + ax + b \ (mod\ p)$ mit $a, b \in \mathbb{Z}_p$ und $-4a^3 - 27b^2 \not\equiv 0 \ (mod\ p)$ erfüllen, zusammen mit dem sogenannten Punkt im Unendlichen ∞.



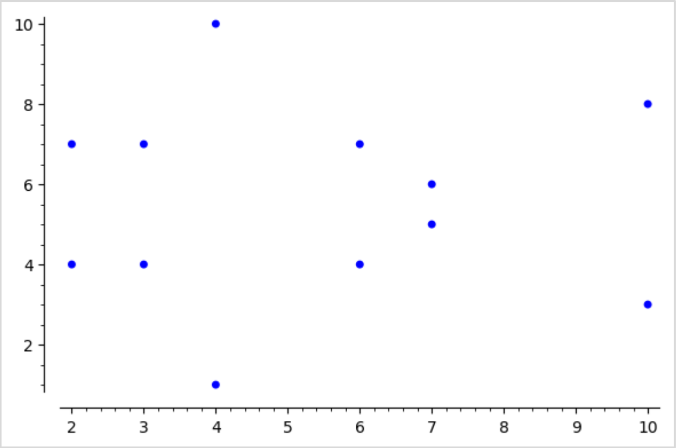
- Addition und skalare Multiplikation funktionieren ähnlich, aber $mod\ p$



- Logarithmus-Problem auf elliptischen Kurven (ECDLP):
  - Finde ein $n \in \mathbb{N}$ für gegebene Punkte $P, Q \in E$ sodass $Q = nP$ gilt.
  - Logarithmus-Problem über $\mathbb{R}$ ist »leicht« zu berechnen, aber wird schwierig über dem endlichen Körper $\mathbb{Z}_p$

https://andrea.corbellini.name/images/singular-mod-p.png
https://andrea.corbellini.name/images/point-addition-mod-p.png

- Beispiel:
  - Elliptische Kurve über $\mathbb{Z}_{11}$ mit $a = 3, b = 2$
  - $E(\mathbb{Z}_{11}) = \{(x, y) \in (\mathbb{Z}_{11})^2 \mid y^2 = x^3 + 3x + 2\} \cup \{\infty\}$



  - Punkte der Kurve: $\{ (2,4),\ (2,7),\ (3,4),\ (3,7),\ (4,1),\ (4,10),\ (6,4),\ (6,7),\ (7,5),\ (7,6),\ (10,3),\ (10,8),\ \infty \}$
  - Generator der Gruppe: $P = (2,7), 2P = (10,8), 3P = (4,1), \ldots, 13P = \infty$

Elliptic Curve Cryptography (ECC)

- Anwendung von elliptischen Kurven für Kryptographie
  - Nicht alle elliptischen Kurven sing gleich gut für ECC geeignet.
  - Ordnung (Punkteanzahl) von E(K) ist u.a. wichtig für Sicherheit.
  - Methoden zum Finden geeigneter elliptischer Kurven:
    - Wähle a, b, p, berechne die Gruppenordnung und überprüfe Sicherheit
    - Wähle p und Gruppenordnung und bestimme a und b (schnellere Methode)
- Vorteile
  - bei vergleichbarem Sicherheitsniveau in deutlich kürzere Schlüssel (≥ 200 Bit)
  - erzeugen deutlich kürzere Signaturen
  - weniger Rechenaufwand
- Beispiele für Algorithmen
  - ECDH: Diffie-Hellman-Schlüsselaustausch auf Basis elliptischen Kurven
  - ECDSA: Standardisiertes Signatursystem auf Basis elliptischer Kurven
  - **EC-ElGamal**: ElGamal auf Basis elliptischer Kurven

Public Key Infrastructures

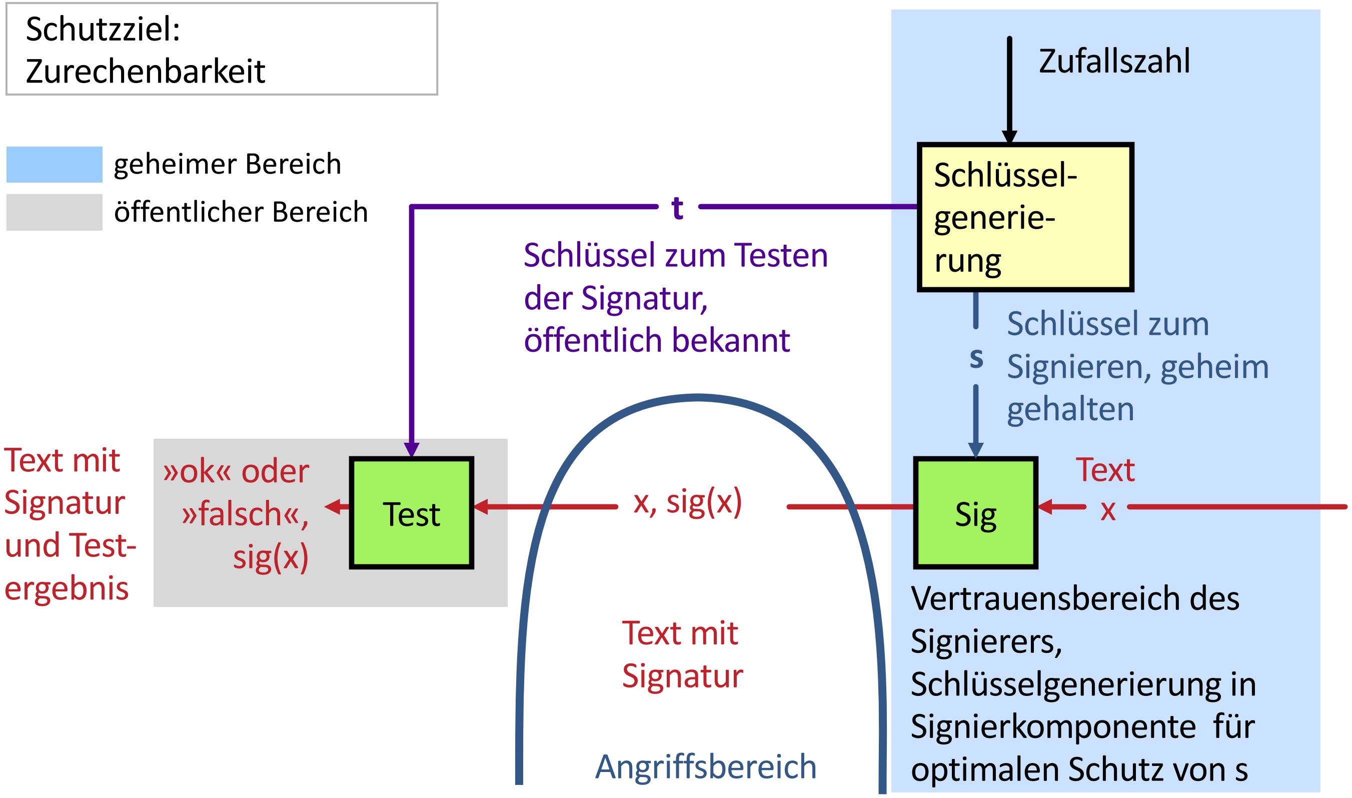
Signierte Nachricht und Zertifikat

```
-----BEGIN SIGNED MESSAGE-----

Hiermit bestelle ich folgende Waren:

 10 Eier                 Euro  2,00
  1 Flasche Milch        Euro  1,50
  1 Kasten Bier          Euro 15,00
 -------------------------------
 Gesamtbetrag            Euro 18,50

Die Zahlung erfolgt bei Lieferung.

Hannes Federrath

-----BEGIN SIGNATURE-----
iQAAwUBOi9VLoBzbJXQK0fCYo1rMKCKrdhAn2Rs
amogkkm+Off90L0W5RxUubfVuUFSXuv=

-----END SIGNED MESSAGE-----
```

```
-----BEGIN CERTIFICATE-----

Name: Hannes Federrath

Public key:
h833hd38dddajscbicme098k236egfkw74h5445
84hdbscldmrtpofjrkt0jedagaszw12geb3u4b=

Valid from:  19.11.2014
Valid until: 18.11.2017

Issuer: Einwohnermeldeamt Dresden


-----BEGIN SIGNATURE OF ISSUER-----
23j423vdsaz345kj435ekj4z2983734ijo23i72
kj867wdbez2o074j5lkdmcd1237t3rgbdvbwdj=

-----END CERTIFICATE-----
```

Digitales Signatursystem

Schutzziel:
Zurechenbarkeit



Signaturgesetz (SigG) vom 16. Mai 2001 schafft rechtliche Rahmenbedingungen für den Beweiswert digitaler Signaturen

Elektronische Signatur

Daten in elektronischer Form, die

- anderen elektronischen Daten beigefügt oder logisch mit ihnen verknüpft sind und die zur Authentifizierung dienen

Fortgeschrittene Signatur

Daten in elektronischer Form, die

- ausschließlich dem Signaturschlüssel-Inhaber zugeordnet sind
- die Identifizierung des Signaturschlüssel-Inhabers ermöglichen
- mit Mitteln erzeugt werden, die der Signaturschlüssel-Inhaber unter seiner alleinigen Kontrolle halten kann
- mit den Daten, auf die sie sich beziehen, so verknüpft sind, dass eine nachträgliche Veränderung der Daten erkannt werden kann

Qualifizierte Signatur

Daten in elektronischer Form, die

- die Anforderungen an eine fortgeschrittene Signatur erfüllen
- auf einem zum Zeitpunkt ihrer Erzeugung gültigen qualifizierten Zertifikat beruhen
- mit einer sicheren Signaturerstellungseinheit erzeugt werden

Sicherheit

Signaturgesetz (SigG) vom 16. Mai 2001 schafft rechtliche Rahmenbedingungen für den Beweiswert digitaler Signaturen

Elektronische Signatur

Beispiel: **E-Mail mit »Signatur«**

```
From: Hannes Federrath
Subject: Beispiel

Das ist der Text.
--
Hannes Federrath
FB Informatik
Uni Hamburg
```

Fortgeschrittene Signatur

Beispiel: **PGP-signierte E-Mail**

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Das ist der Text.

-----BEGIN PGP SIGNATURE-----
Version: PGP 8.0.2

iQA/AwUBP6wDdOFAIGFJ7x2EEQK9VgCg2Q4
eQAztVIHP0HNFQ10eaXte96sAnR2p
53T/SdevjXIuX6WOF5IXA44S
=K3TO
-----END PGP SIGNATURE-----
```

Qualifizierte Signatur

Zertifikatausstellung nach Identitätsüberprüfung

sichere Signaturerstellungseinheit

Sicherheit

Zweck der Schlüsselzertifizierung

- Betrifft öffentlichen Testschlüssel für die digitale Signatur
- Zertifikat:
  - bestätigt die Zusammengehörigkeit von Testschlüssel und Benutzeridentität bzw. Testschlüssel und Pseudonym.
  - Enthält selbst die Signatur des Zertifizierers
- Ohne Zertifikate:
  - Angreifer kann ein Schlüsselpaar generieren und einfach behaupten, dass dieser Schlüssel jmd. gehört.
  - Testschlüssel sind wertlos ohne Zertifikat (zumindest in einer offenen Welt)

Maskerade-Angriff 1/2

Alice hat Schlüsselpaar generiert und will ihn veröffentlichen.

Alice <alice@abc.de>

-----BEGIN PGP PUBLIC KEY BLOCK-----
mQGiBDQyJk0RBADVPjcdvwmyOtqsZBt6z4/5M9MYDB
i+dYN QEBXQcH/RGe2i30LRvRk4asX++JSTylku
8LMOl W+lbmsVNXeQSdmbSAUfd3d9bI/+fGwQcz
6W8l kfDaF7xPI7oVZUY1I7cqEfTvic003bgL
sUZ ukKj01O66wVmqlnXcbi2XUebka
LO ZW59gf5I0eUBevSmydIaliH9Pm
---- END IC KEY BLOCK-----

c_{Alice}

Angreifer

- hält c_{Alice} zurück (blockiert Verteilung)
- generiert selbst ein Schlüsselpaar c_{Mask}, d_{Mask} unter falschem Namen
- schickt c_{Mask} an Bert

c_{Mask}

Bert besitzt jetzt nicht authentischen Schlüssel von Alice.

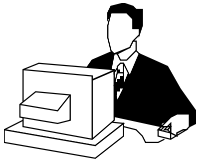
Alice <alice@abc.de>

-----BEGIN PGP PUBLIC KEY BLOCK-----
OTUAoLncfli6Yit0Kqgp/N9h37uopJHbiQCVAw
xBBPLRdmalP22ij0dARxbJLO7u7XOrnyV3b
l4ydps/ruj9yaY62BwQNMEoGjAnZGA5t3MM
7ZLp1dmFYYVYPL4xRfOJ+MF5ifb8RXaDAl+
CwMBAgAKCRDhQCBhSe8dhOYYAJsE
u64hbO2wuFQlwwq1yb+JAD8DBRA0
-----END PGP PUBLIC KEY BLO

Maskerade-Angriff 2/2

Ohne die Gewissheit über die Echtheit eines öffentlichen Schlüssels funktioniert keine sichere asymmetrische Kryptographie. Deshalb: Schlüsselzertifizierung

Bert will Alice eine Nachricht N schicken.



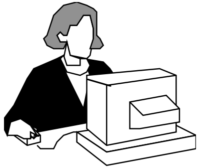
$c_{Mask}(N)$

Angreifer:

- Weiterleitung verhindern
- entschlüsseln von $c_{Mask}(N)$ mit d_{Mask}
- verschlüsseln von N mit c_{Alice}

$c_{Alice}(N)$

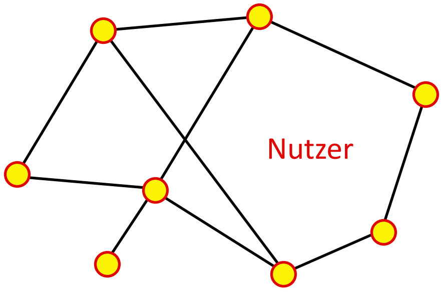
Alice erhält die Nachricht N.
N ist verschlüsselt mit ihrem öffentlichen Schlüssel.



Zertifizierungsmodelle

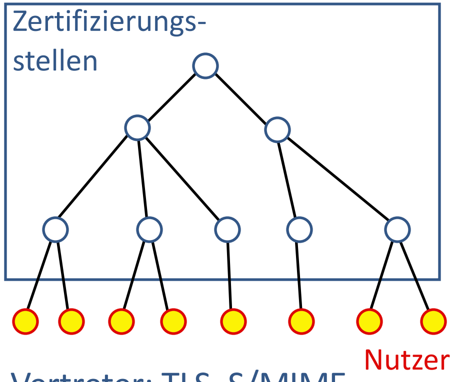
- Zur Verschlüsselung und Signierung wird asymmetrische Kryptographie verwendet
  - Zwei Schlüssel: Private und Public Key
  - Zwei Ansätze zur Zuordnung des Public Keys zu einer Person
    - PGP und GnuPG: »Web of Trust«
    - TLS und S/MIME: Hierarchische Zertifizierung

Web of Trust



Vertreter: PGP, GnuPG

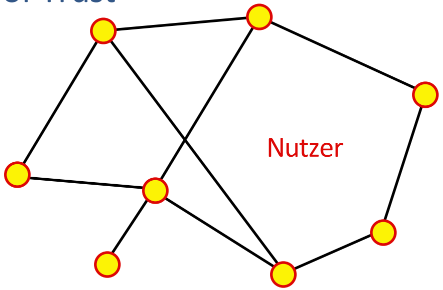
Hierarchische Zertifizierung



Vertreter: TLS, S/MIME

Zertifizierungsmodelle

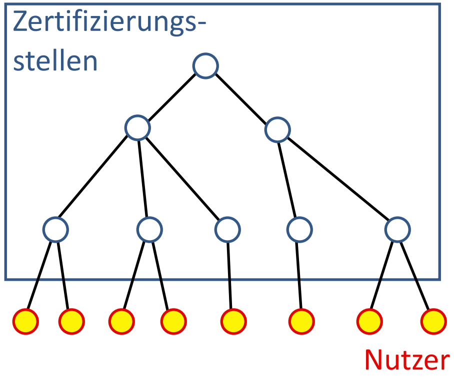
- Web of Trust



Vorteile:
- einfache, flexible Nutzung
- viele potentielle Zertifikatsketten

Nachteile:
- keine oder nur schwer erreichbare Beweisführung im Streitfall
- finden eines vertrauenswürdigen Pfades aufwendiger

- Hierarchische Zertifizierung

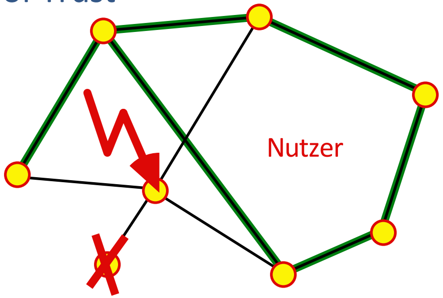


Vorteile:
- klare Strukturen und Zurechenbarkeiten (wichtig im Streitfall)

Nachteile:
- Overhead durch Organisationsstruktur

Zertifizierungsmodelle

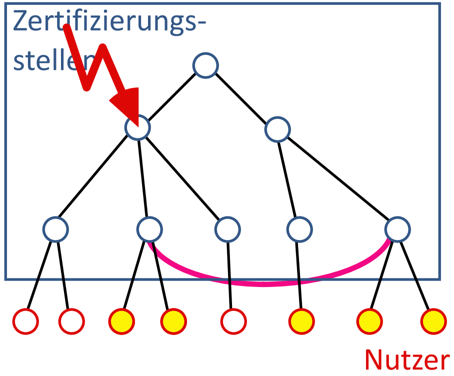
- Web of Trust



Vorteile:
- einfache, flexible Nutzung
- viele potentielle Zertifikatsketten

Nachteile:
- keine oder nur schwer erreichbare Beweisführung im Streitfall
- finden eines vertrauenswürdigen Pfades aufwendiger

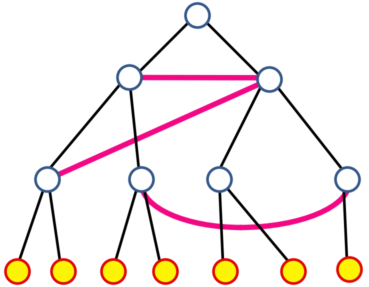
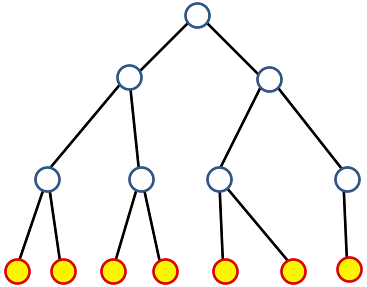
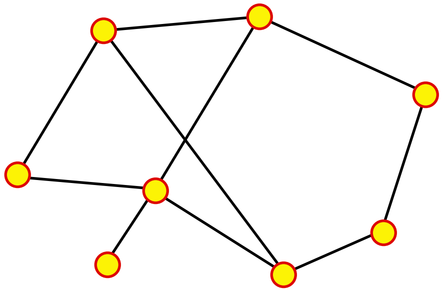
- Hierarchische Zertifizierung



Vorteile:
- klare Strukturen und Zurechenbarkeiten (wichtig im Streitfall)

Nachteile:
- Overhead durch Organisationsstruktur
- anfällig gegen Fehlverhalten
  - Lösungsansatz: Cross Certification erhöht Verfügbarkeit

Zusammenfassung: Zertifizierungsmodelle

Haftung des Zertifizierers	Vermaschter Graph	Baum (minimal zusammenhängender Graph)
Ja	 Reduzieren der Fehleranfäligkeit durch Cross Certification	
Nein		–

X.509-Zertifikate

ITU X.509 Zertifikate

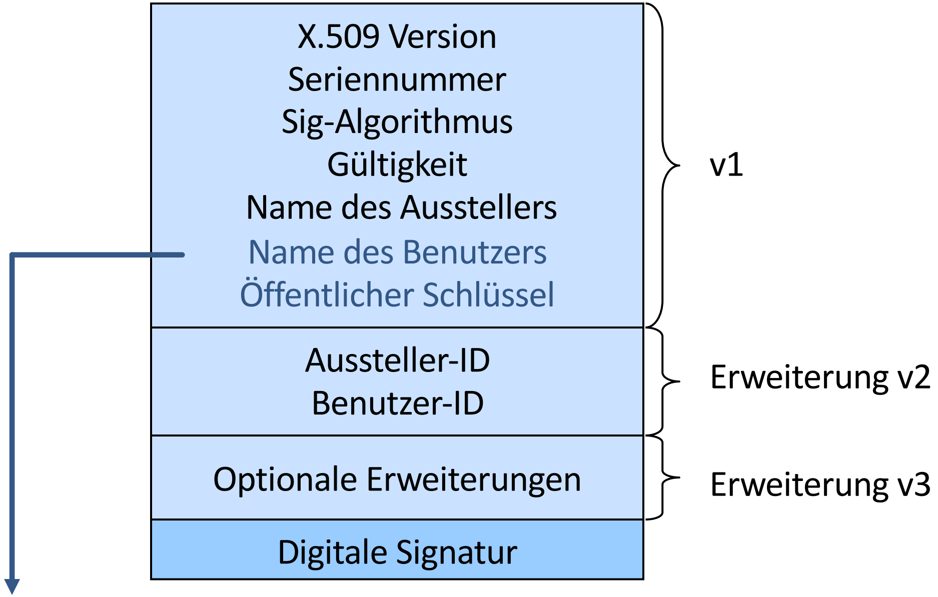
… werden insb. angewendet bei TLS und S/MIME.

- S/MIME (Secure Multipurpose Internet Mail Extensions)
  - Ursprünglich von RSA Data Security Inc.
  - S/MIME v3 im Juli 1999 als IETF-Standard verabschiedet
  - Internet Standards RFCs 2632-2634 (und weitere)
  - In die meisten E-Mail-Clients integriert
- TLS Transport Layer Security
  - vormals SSL (Secure Sockets Layer)
  - Verschlüsselung von TCP-Verbindungen
  - ursprünglich von Netscape für Browser entwickelt
  - heute in jedes moderne Betriebssystem integriert

Schutz der Vertraulichkeit und Integrität

ITU X.509 Zertifikate

- Festlegung eines standardisierten Formats für Zertifikate



Hierarchisch aufgebauter »distinguished name«, z.B.:
cn = Hannes Federrath, ou = Informatik, o = Uni Hamburg, c = DE

ITU X.509 Zertifikate

- Erweiterungen in X.509v3
  - Art des Schlüssels, Anwendungsbereich
  - Alternative Namen für Inhaber und Aussteller
  - Einschränkungen bzgl. cross certification
  - Informationen bzgl. Sperrlisten (URL)
  - private ausstellerspezifische Erweiterungen
- Zertifizierungsprozess
  - Antrag bei der Registration Authority (RA)
  - Identitätsprüfung durch die RA
  - Zertifizierung durch die Certificate Authority (CA)
  - Ausgabe des Zertifikats an den Antragsteller
- Widerruf der Zertifikate
  - Certificate Revocation Lists (CRLs)
  - Online Certificate Status Protocol (OCSP)

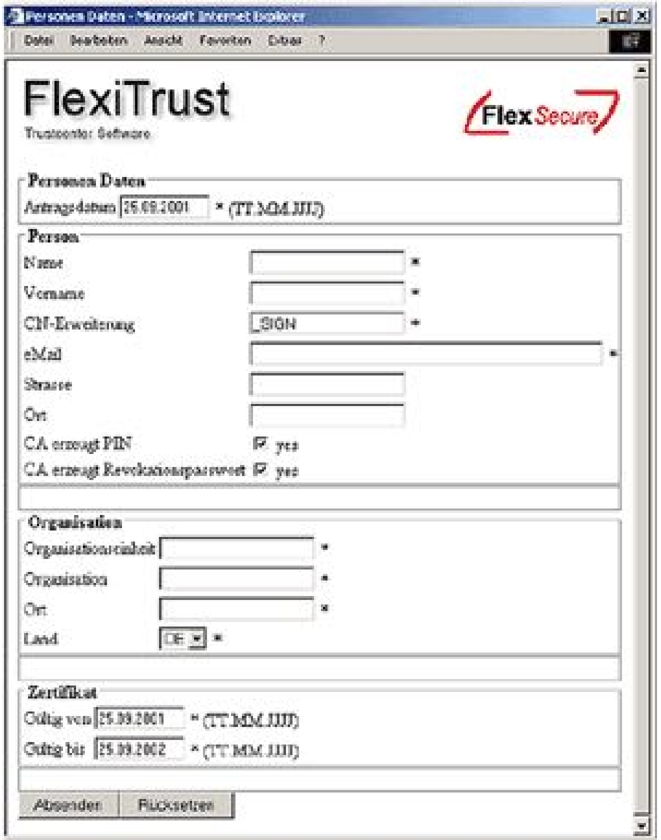
Optionen für die Zertifikatserstellung

- OpenCA
  - http://www.openca.org
  - Beschreibung:
    »Open Source out-of-the-box Certification Authority implementing the most used protocols with full-strength cryptography world-wide.«
  - Benutzt OpenLDAP, OpenSSL, Apache, mod_ssl, Perl und lässt sich per Browser bedienen/konfigurieren
  - Antragstellung per Web-Browser bei der RA, nach Zertifizierung durch die CA Download des Zertifikats
  - Kosten:
    - Potentiell niedrig
  - Administrationsaufwand:
    - Potentiell mittel
    - allerdings hoher Einrichtungsaufwand

Optionen für die Zertifikatserstellung

- FlexiTrust
  - http://www.flexsecure.de (jetzt Kobil)
  - Umfassende Lösung zur Erstellung einer PKI und den Betrieb eines Trustcenters
  - Entwicklung der Technischen Universität Darmstadt
  - Nach dem Signaturgesetz zertifiziert
  - Plattformunabhängige Java-Anwendung

  - Kosten:
    - Sehr hoch
  - Administrationsaufwand:
    - Potentiell niedrig



Bezug von X.509-Zertifikaten von einem kommerziellen Anbieter

- Verschiedene Zertifikatsklassen
  - Class 0: Demo-Zertifikat
  - Class 1: Existenz der E-Mail-Adresse wird geprüft
  - Class 2: Schriftlicher Auftrag
  - Class 3: Antragsteller muss sich persönlich identifizieren
  - Class 4: »Online business transactions between companies«
  - Class 5: »for private organizations or governmental security«
- Vorteile:
  - Wurzelzertifikat ist in den meisten Clients schon enthalten
  - keine eigene CA nötig
  - kein Administrationsaufwand
- Kosten: 0-130 EUR pro Jahr und Zertifikat

TLS-Handshake

Client

Server-Zertifikat prüfen
- Common Name ist gleich Domain Name
- Gültigkeit
- Signatur des Issuers

Zertifikatskette prüfen
- CA-Attribut
- Gültigkeit
- Signatur des Issuers

ab hier symmetrisch verschlüsselt

ClientHello →

← ServerHello

← Certificate

← ServerKeyExchange

← CertificateRequest

← ServerHelloDone

Certificate →

ClientKeyExchange →

CertificateVerify →

ChangeCipherSpec →

Finished →

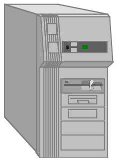
← ChangeCipherSpec

← Finished

← Application Data →

Server

server.com

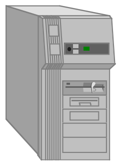


ab hier symmetrisch verschlüsselt

TLS-Handshake (vereinfacht)

Client ruft auf
https://server.com/

Server
server.com



ClientHello →

← ServerHello

← Certificate

← ServerKeyExchange

← CertificateRequest

← ServerHelloDone

Certificate →

ClientKeyExchange →

KeyExchange →

CertificateVerify →

ChangeCipherSpec →

Finished →

← ChangeCipherSpec

← Finished

← Application Data →

Server-Zertifikat prüfen
- Common Name ist gleich Domain Name
- Gültigkeit
- Signatur des Issuers

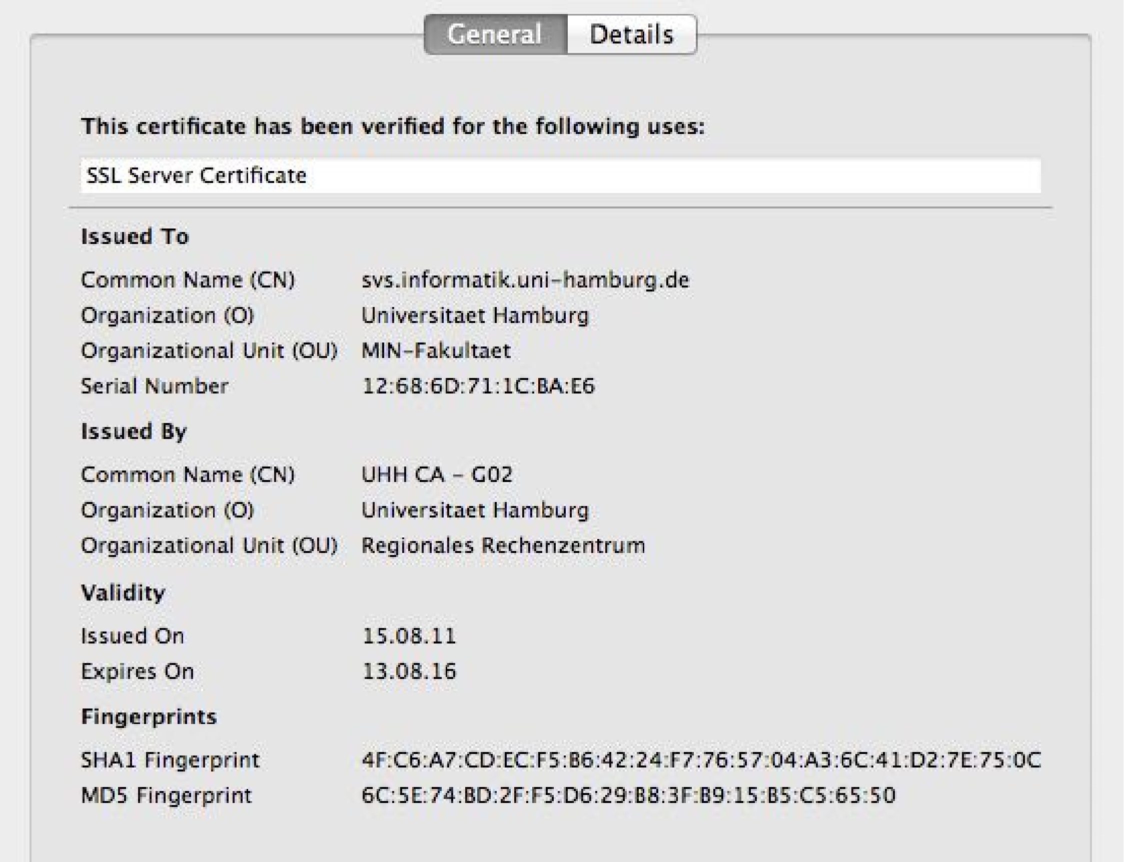
Zertifikatskette prüfen
- CA-Attribut
- Gültigkeit
- Signatur des Issuers

Client teilt dem Server den symmetrischen Schlüssel mit (verschlüsselt mit Public Key des Servers)

ab hier symmetrisch verschlüsselt

ab hier symmetrisch verschlüsselt

Benutzeransicht eines Zertifikats (Firefox)

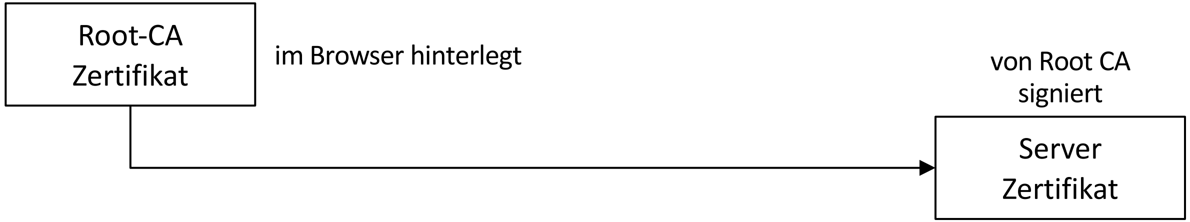


Normales Serverzertifikat

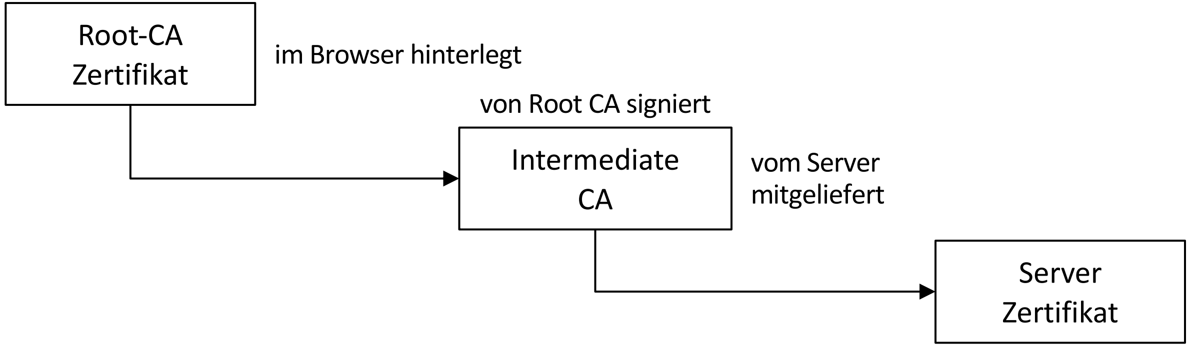
Extended Validation Certificate

Zertifikatsvalidierung: Root CA vs. Intermediate CA

- Fall 1: Root CA stellt direkt ein Server-Zertifikat aus

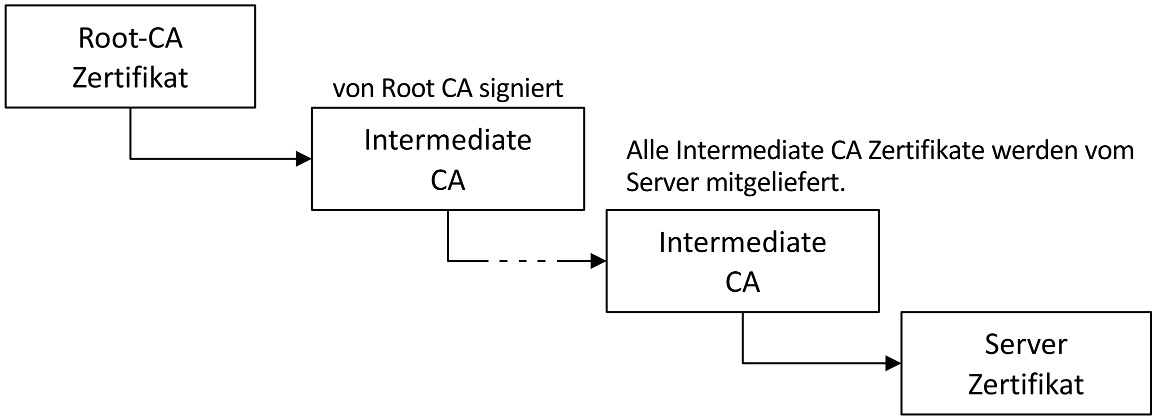


- Fall 2: Root CA zertifiziert Intermediate CA, diese stellt Server-Zertifikat aus



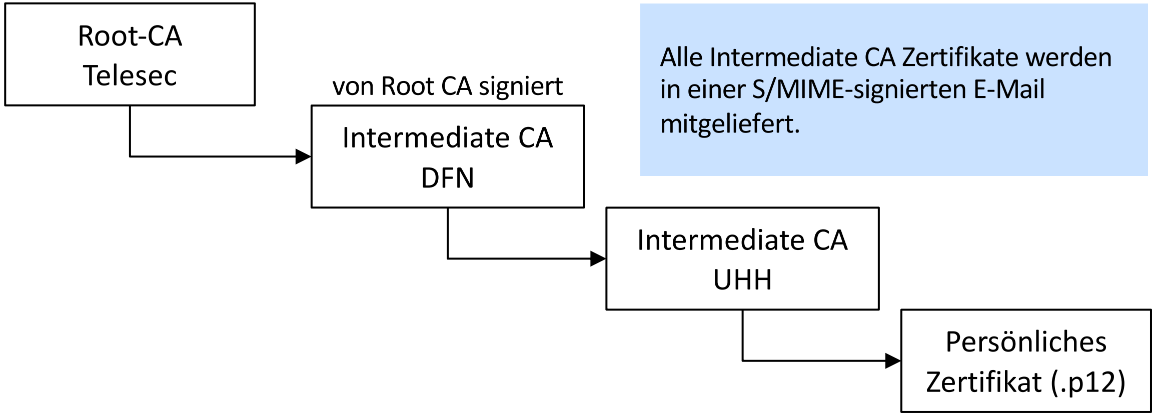
Rekursive Zertifikatsvalidierung

- Rekursives Verfahren:
  - Root CA zertifiziert Intermediate CA, Intermediate CA zertifiziert Intermediate CA, u.s.w, Intermediate CA stellt ein Server-Zertifikat aus



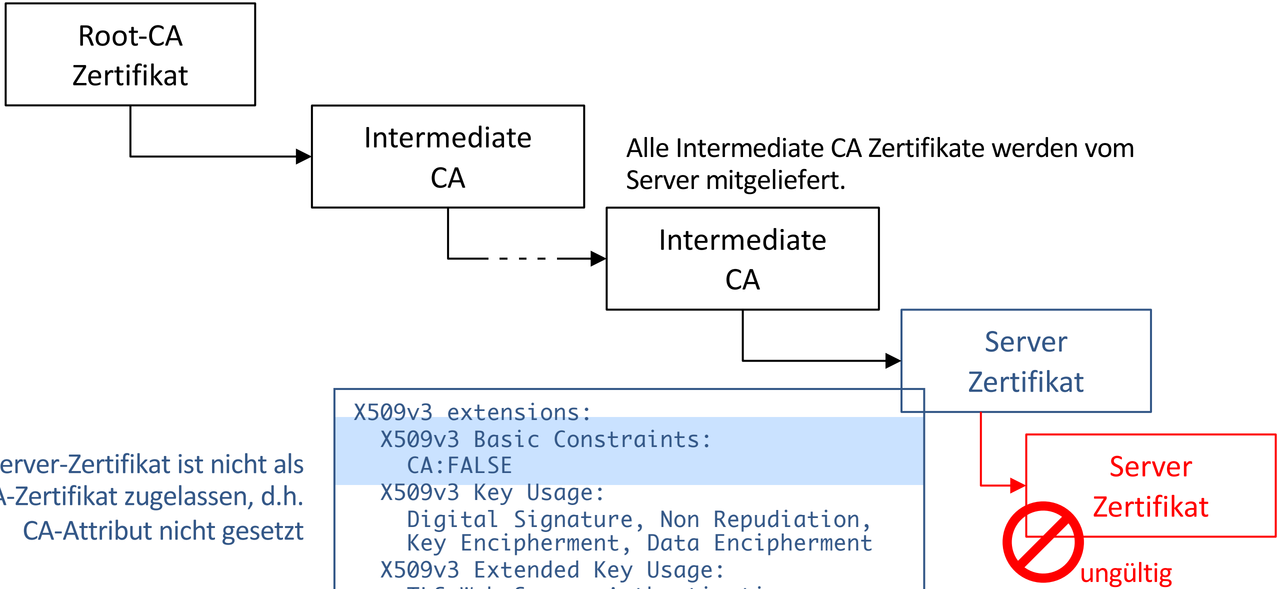
Rekursive Zertifikatsvalidierung: Beispiel S/MIME-Zertifikate der UHH

- Rekursives Verfahren:
  - Root CA zertifiziert Intermediate CA, Intermediate CA zertifiziert Intermediate CA, Intermediate CA stellt ein Server-Zertifikat aus



Einschränkungen bzgl. Cross Certification

Wie wird verhindert, dass vom Inhaber des Server-Zertifikats ein weiteres gültiges Server-Zertifikat erzeugt wird?



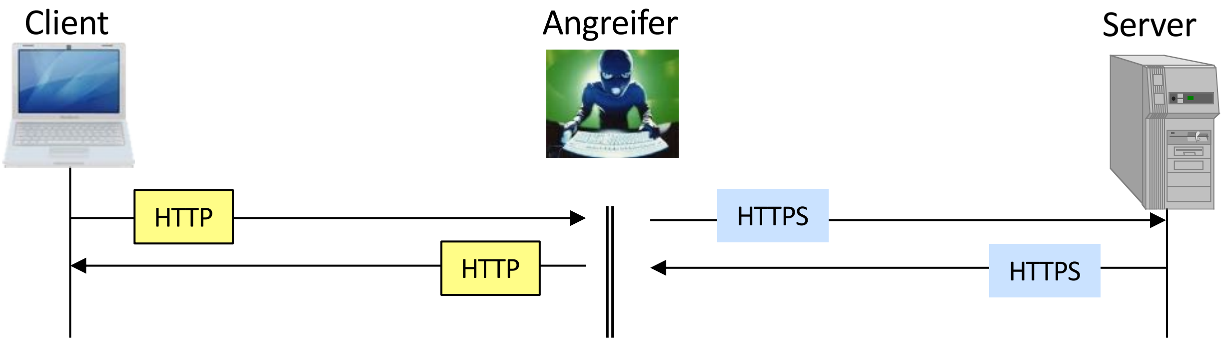
Alle Intermediate CA Zertifikate werden vom Server mitgeliefert.

Server-Zertifikat ist nicht als CA-Zertifikat zugelassen, d.h. CA-Attribut nicht gesetzt

```
X509v3 extensions:
  X509v3 Basic Constraints:
    CA:FALSE
  X509v3 Key Usage:
    Digital Signature, Non Repudiation,
    Key Encipherment, Data Encipherment
  X509v3 Extended Key Usage:
    TLS Web Server Authentication
```

Man-in-the-Middle-Angriffe auf HTTPS: sslstrip

- Ziel: Angreifer möchte Kommunikation mitlesen und/oder verändern
- Angriffsmethode: Angreifer verhindert Umleitung zu HTTPS



Nutzer geben www.bank.de ein — ohne https://

Angreifer verhindert Umleitung zu HTTPS

Server liefert normalerweise:

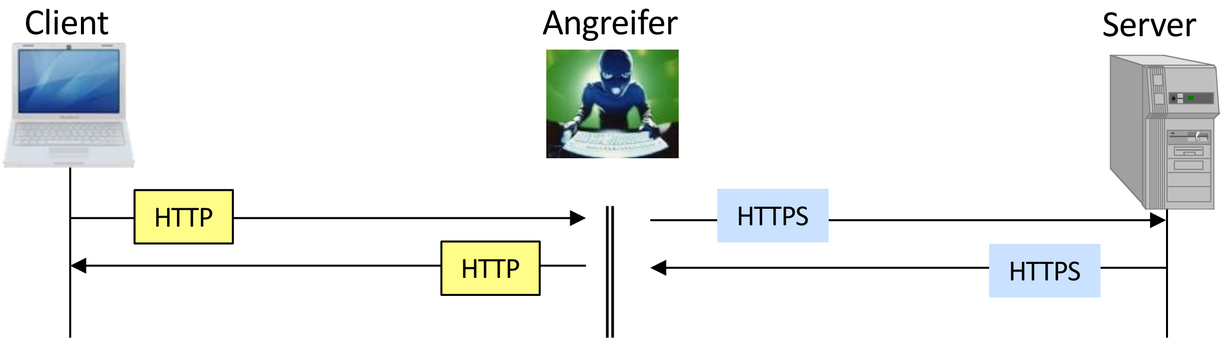
```
> GET / HTTP/1.1
> Host: www.bank.de

< HTTP/1.1 301 Moved Permanently
< Location: https://www.bank.de/
```

HTTP Strict Transport Security (HSTS): HTTP-Header liefert bei Erstkontakt Verfallsdatum für https-Zwang – funktioniert nur, wenn Vorkontakt zum Server ohne Angreifer

```
Strict-Transport-Security: max-age=<sek>
```

Man-in-the-Middle-Angriffe auf HTTPS: sslstrip

- Ersetzen aller Umleitungen und Links zu HTTPS
- Verbindung zum Server per HTTPS, zum Client per HTTP
- Server merkt nicht, dass Client kein HTTPS verwendet



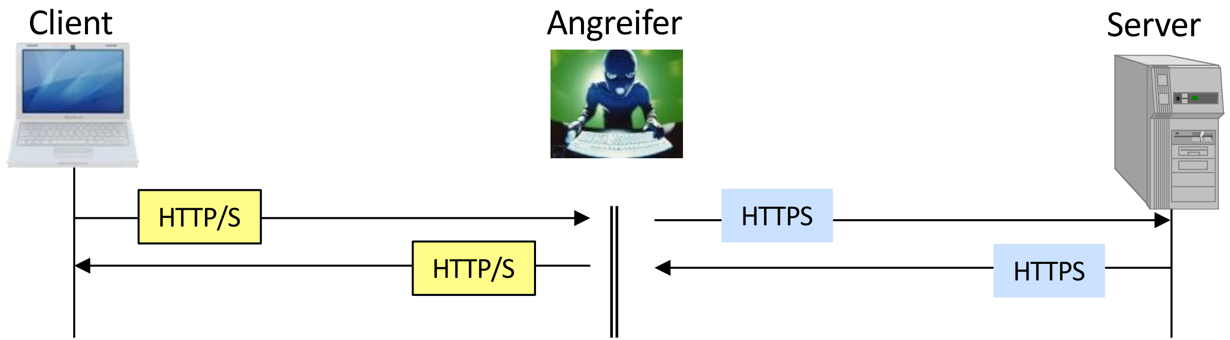

```
<html> [...]
<body> [...]
<a href="http://www.shop.de/login">
Zum sicheren Login-Formular</a>[...]
</body></html>
```

wird vom Angreifer ersetzt durch

```
<html> [...]
<body> [...]
<a href="https://www.shop.de/login">
Zum sicheren Login-Formular</a>[...]
</body></html>
```

Man-in-the-Middle-Angriffe auf HTTPS: burp proxy, mitmproxy

- Ziel: Angreifer möchte Kommunikation mitlesen und/oder verändern
- Angreifer stellt »on the fly« gültige Zertifikate aus



Nutzer haben etwa Firmen-CA-Zertifikat importiert (Angreifer ist Mitarbeiter der Firma)

»On the fly« vom Angreifer ausgestellte Zertifikate

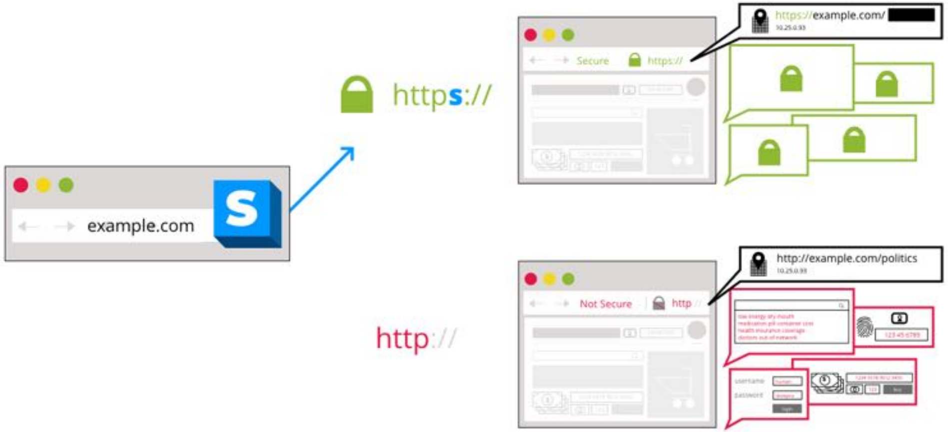
HTTP Public Key Pinning (HPKP): HTTP-Header liefert Hash des korrekten öffentlichen Schlüssels – funktioniert nur, wenn Vorkontakt zum Server ohne Angreifer

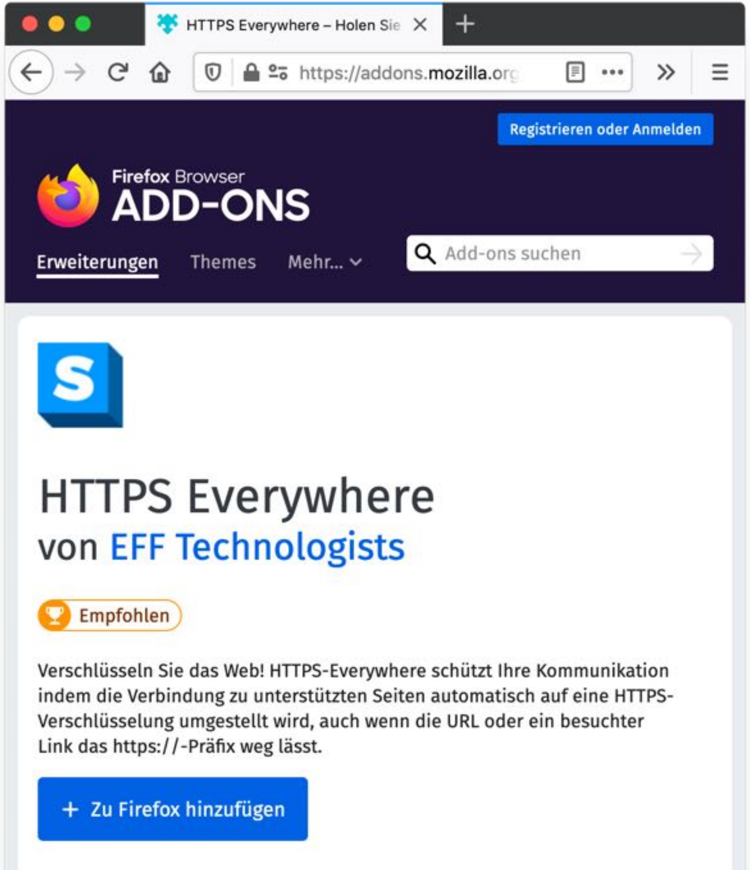
```
Public-Key-Pins: pin-sha256="cUPcTAZWK..."; max-age=<sek>
```

HTTPS Everywhere

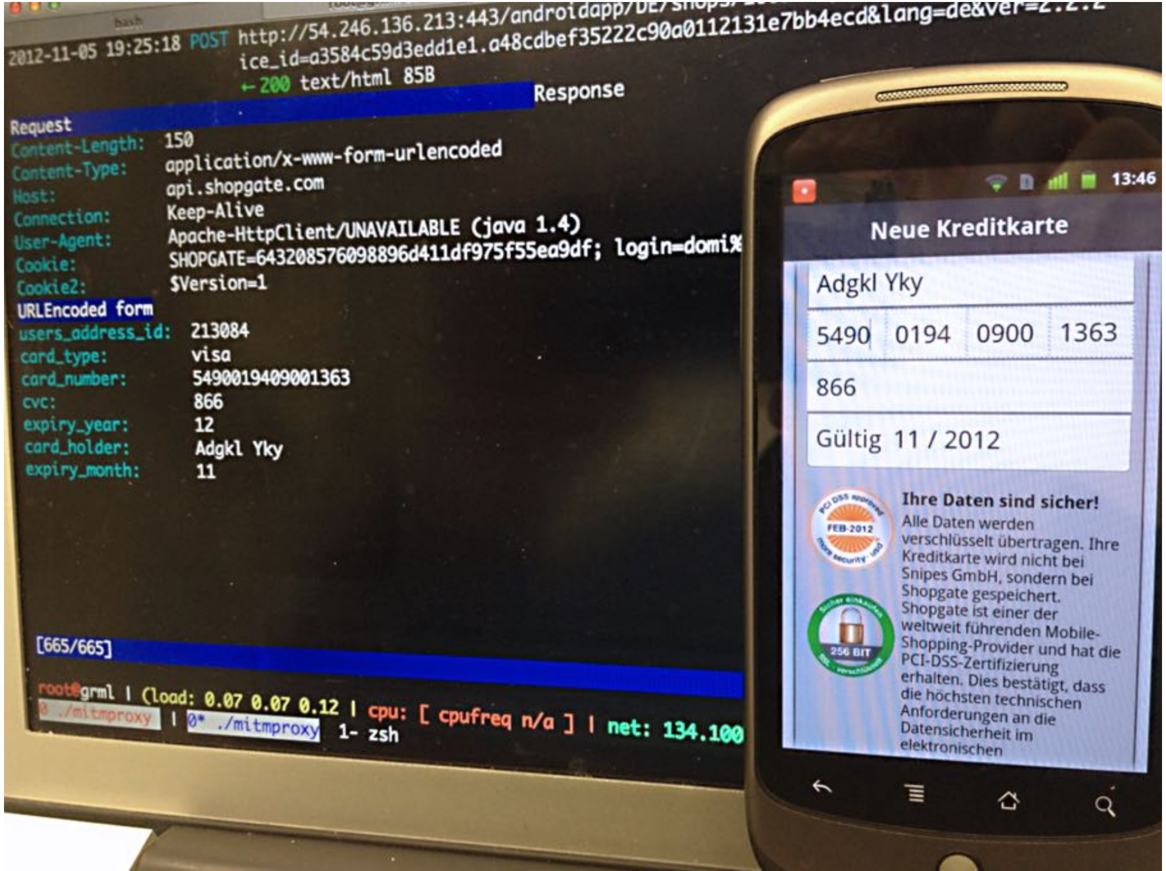
https://addons.mozilla.org/de/firefox/addon/https-everywhere/

- URLs werden um den https-Präfix ergänzt
- Erweiterung für Firefox und Chrome





Fehlende oder fehlerhafte Zertifikatsvalidierung in Clients



Nach welchen Kriterien kommen die Root-Zertifikate in Anwendungen wie Firefox oder die Betriebssysteme?

- Audit-Standards
  - Überprüfung der Steuerungs- und Kontrollprozesse von (Root)-CAs

Plattform	**Anzahl Zertifikate**
Apple iOS, OS X	150
Microsoft Windows 10	230
Mozilla Network Security Services	130

- WebTrust-Standard der Canadian Instititute of Chartered Accountants
- ETSI TS 102 042 – Policy for requirements for certification authorities – Issuing public key certificates
- ETSI TS 101 456 – Policy for requirements for certification authorities – Issuing qualified certificates
- ISO 21128 – Public key infrastructure for financial services – Practices and policy framework

Quelle: Hicken 2017

Nach welchen Kriterien kommen die Root-Zertifikate in Anwendungen wie Firefox oder die Betriebssysteme?

- Audit-Standards
  - Überprüfung der Steuerungs- und Kontrollprozesse von (Root)-CAs

	Anzahl Zertifikate	
Plattform	Hicken 2017	2020
Apple iOS, OS X	150	217
Microsoft Windows 10	230	392
Mozilla CA Certificate Store	130	150
Google (Android)		135 (in 2018)

- WebTrust-Standard der Canadian Instititute of Chartered Accountants
- ETSI TS 102 042 – Policy for requirements for certification authorities – Issuing public key certificates
- ETSI TS 101 456 – Policy for requirements for certification authorities – Issuing qualified certificates
- ISO 21128 – Public key infrastructure for financial services – Practices and policy framework

Biometrischer Reisepass

Ein Beispiel für den internationalen Aufbau einer PKI

Elektronischer Reisepass: RFID zur drahtlosen Kommunikation

Biometrische Daten in elektronischen Reisepässen

- Basic Access Control und Active Authentication

optisches Lesen der MRZ

RF-Chip kennt K (56 Bit)

Aktivierung durch elektr. Feld abwarten, Zufallszahl Rand1 wählen und senden

Response entschlüsseln, Rand1 prüfen, k_{Reader} zwischenspeichern, Schlüsselhälfte k_{Chip} wählen, Response (3-DES) auf Rand2 bilden: $K(Rand1,Rand2,k_{Chip})$ und senden

Elektrisches Feld

Proximity coupling <=10cm

Rand1

$K(Rand1,Rand2,k_{Reader})$

$K(Rand1,Rand2,k_{Chip})$

Alle weitere Kommunikation wird mit (k_{Reader},k_{Chip}) verschlüsselt (3-DES mit CBC)

Leser bildet Schlüssel K (56 Bit) aus MRZ: K=h(PassID,GebD,GültD), aktiviert elektr. Feld und wartet Empfang von Rand1 ab

wählt Zufallszahl Rand2, Schlüsselhälfte k_{Reader} und bildet Response (3-DES) auf Rand1: $K(Rand1,Rand2,k_{Reader})$ und sendet

Response entschlüsseln, Rand2 prüfen

Machine Readable Zone (MRZ):

```
P<D<<NAME<<VORNAMEN<<<<<<<<<<<<<<<
123456789P<<JJMMDDP<JJMMDDP<<<<<<P
```

Passnummer Geburtsdatum Gültigkeitsdatum
jeweils mit Prüfziffer versehen

nach: Dr. Dennis Kügler: Risiko Reisepass? Schutz der biometrischen Daten im RF-Chip. ct 5 (2005) 88

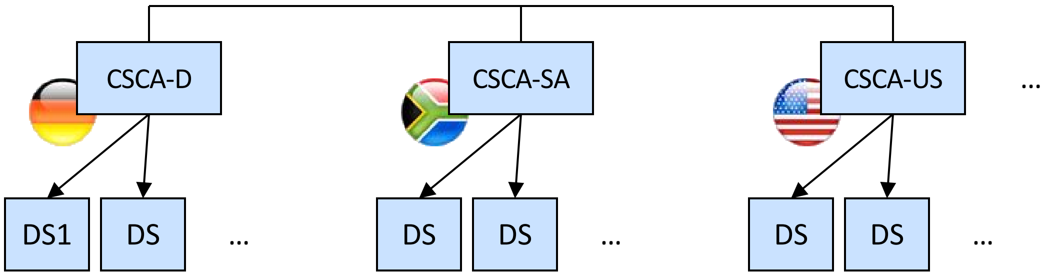
Sicherheitsfunktionen in elektronischen Reisepässen

- Basic Access Control
  - Auslesen der biometrischen Daten benötigt optische Daten der maschinenlesbaren Zone
  - Schutz des digitalen Fotos
- Active Authentication
  - Soll 1:1-Kopien authentischer Daten auf gefälschten Pässen (Chips) verhindern
  - Authentifikation eines Originalchips mittels Challenge-Response

- Symmetrisch verschlüsselte Kommunikation
  - zwischen Pass und Lesegerät
- Passive Authentication
  - Digitale Signatur der gespeicherten Biometriedaten
  - Aufbau der PKI: -->(nächste Folie)

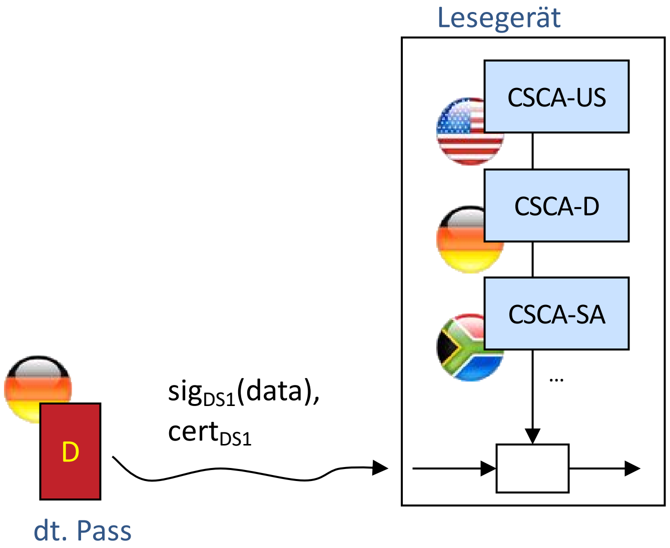
- Technische Spezifikation: http://www.icao.int/mrtd/

- Aufbau der PKI bei Passive Authentication der Biometriedaten
  - eine Country Signing Certifcation Authority (CSCA) pro Land
  - zertifiziert Testschlüssel mehrerer Document Signer (DS)
  - keine übergeordnete weltweite CA
  - alle Lesegeräte enthalten die Zertifikate der CSCAs des eigenen Landes und aller fremden Länder
  - Beispiel:



Quelle: Dennis Kügler, Ingo Naumann: Sicherheitsmechanismen für kontaktlose Chips im deutschen Reisepass.DuD 3 (2007)

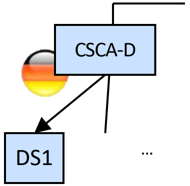
- Aufbau der PKI bei Passive Authentication der Biometriedaten
  - Beispiel: (ausländisches) Lesegerät prüft dt. Passdaten



Vorausgegangen:
- Basic Access Control
- Active Authentication

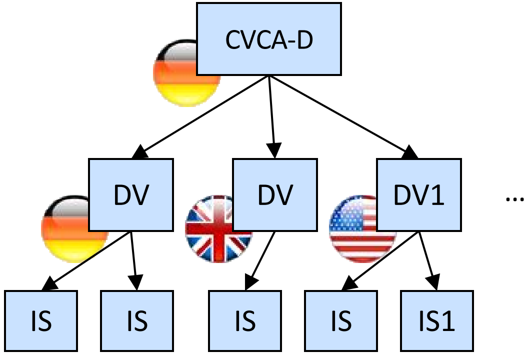
Lesegerät bei Grenzkontrolle:
- kennt alle CSCAs anderer Länder

Ablauf der Zertifikatsprüfung:
- erhält vom Pass: sig_{DS1}(data), $cert_{DS1}$
- prüft: sig_{DS1} mit Testschlüssel (in $cert_{DS1}$ enthalten)
- prüft: $cert_{DS1}$ mit Testschlüssel (in $cert_{CSCA\text{-}D}$)



Sicherheitsfunktionen in elektronischen Reisepässen

- Anstelle von Basic Access Control ab 2007: Extended Access Control
  - Schutz der Fingerabdrücke
  - Beschränkt den Zugriff auf autorisierte Lesegeräte
  - Authentifikation des Lesers über Public Key Zertifikat:
    - Eine Country Verifying Certification Authority (CVCA) pro Land zertifiziert diejenigen
    - Document Verifyer (DV) (fremder Länder), die (Fingerabdruck)-Daten auslesen dürfen.
  - Chips auf Pässen enthalten nationales CVCA-Zertifikat
  - Beispiel:

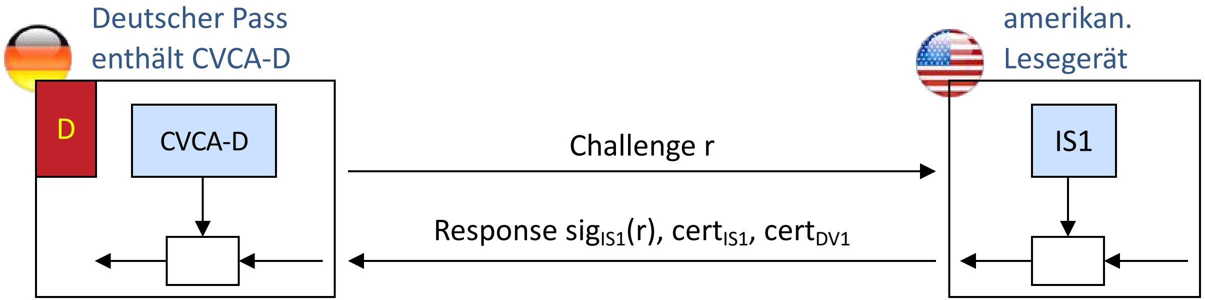


Dt. zertifiziert Public Keys nationaler und internationaler DVs (nur von solchen Ländern, die Fingerabdruckdaten lesen dürfen); DV zertifiziert Public Keys von (landeseigenen) Inspection Systems (IS)

Technische Spezifikation durch BSI: http://www.bsi.de/fachthem/epass/EACTR03110_v101.pdf

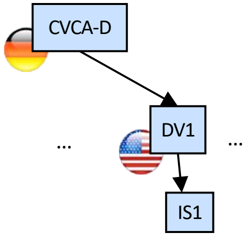
Sicherheitsfunktionen in elektronischen Reisepässen

- Authentifikation des Lesers über Public Key Zertifikat
- Beispiel: dt. Pass prüft amerikan. Leser



Ablauf der Zertifikatsprüfung:

- Pass erhält vom Lesegerät: $sig_{IS1}(r)$, $cert_{IS1}$, $cert_{DV1}$ als Response auf Challenge r (Zufallszahl)
- Pass prüft: sig_{IS1} mit Testschlüssel (in $cert_{IS1}$ enthalten)
- Pass prüft: $cert_{IS1}$ mit Testschlüssel (in $cert_{DV1}$ enthalten)
- Pass prüft: $cert_{DV1}$ mit Testschlüssel (in $cert_{CVCA\text{-}D}$ enthalten)

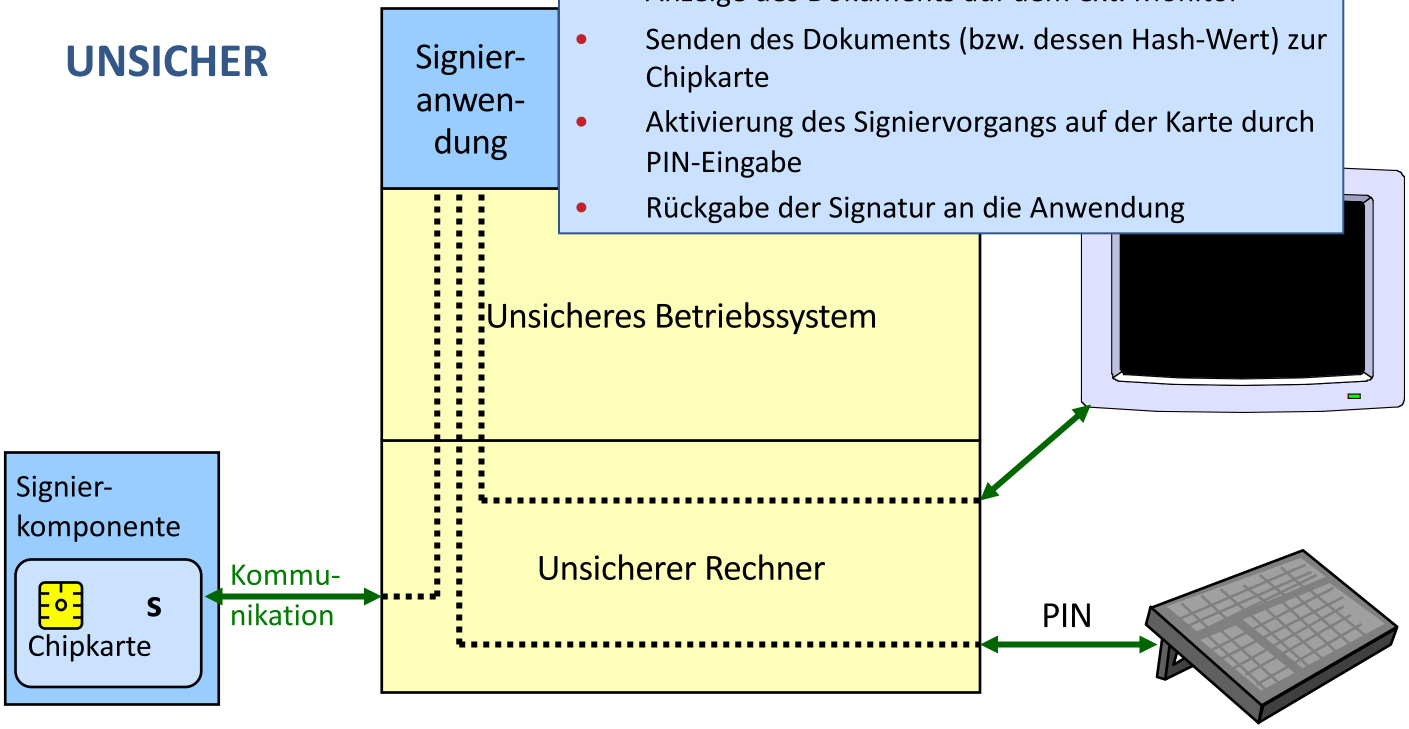


Zur Notwendigkeit sicherer Signaturerstellungseinheiten

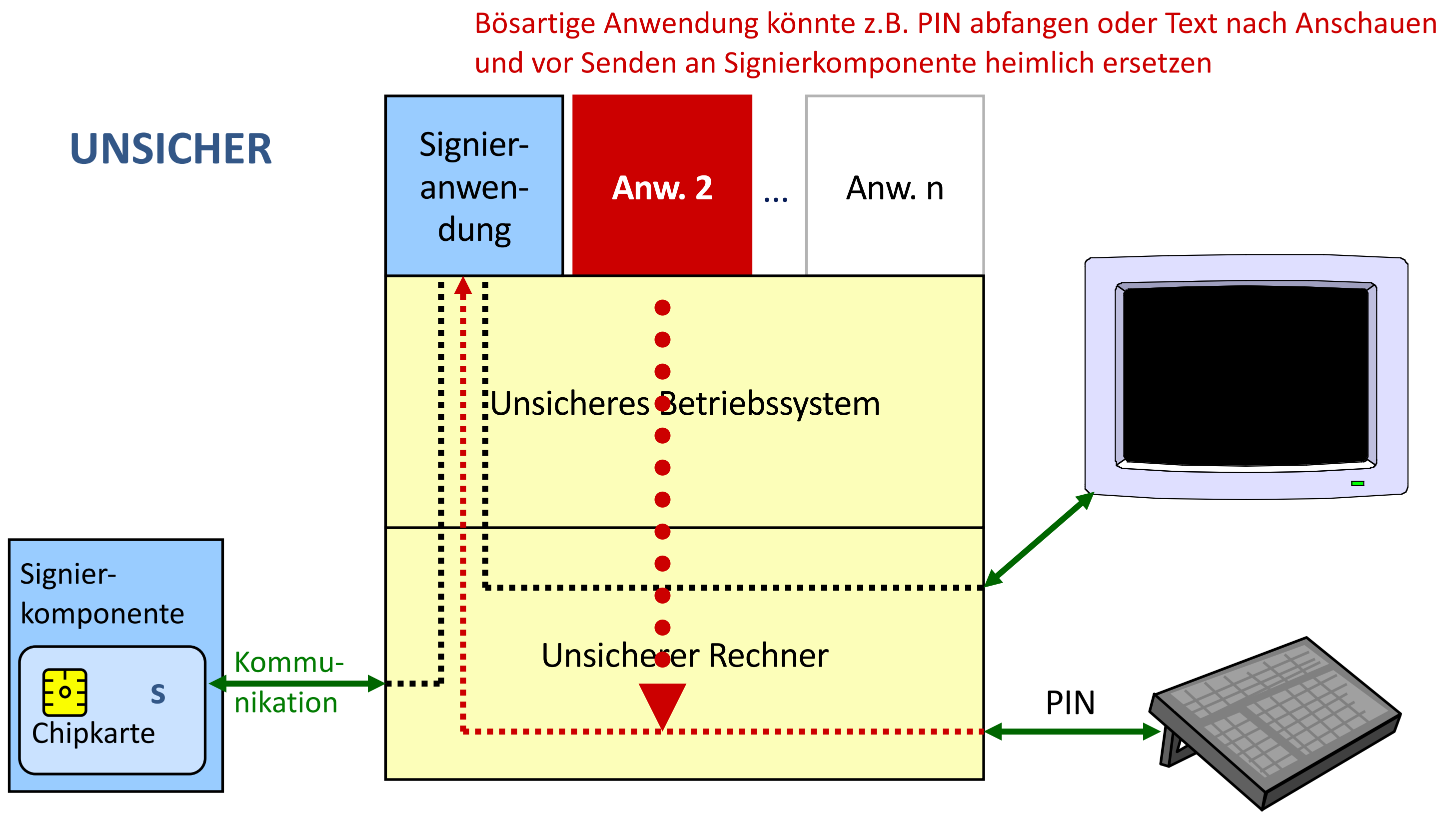
Ablauf auf Standard-PC mit Chipkarte

- Sichere Geräte sind eine Voraussetzung für sichere Signaturen

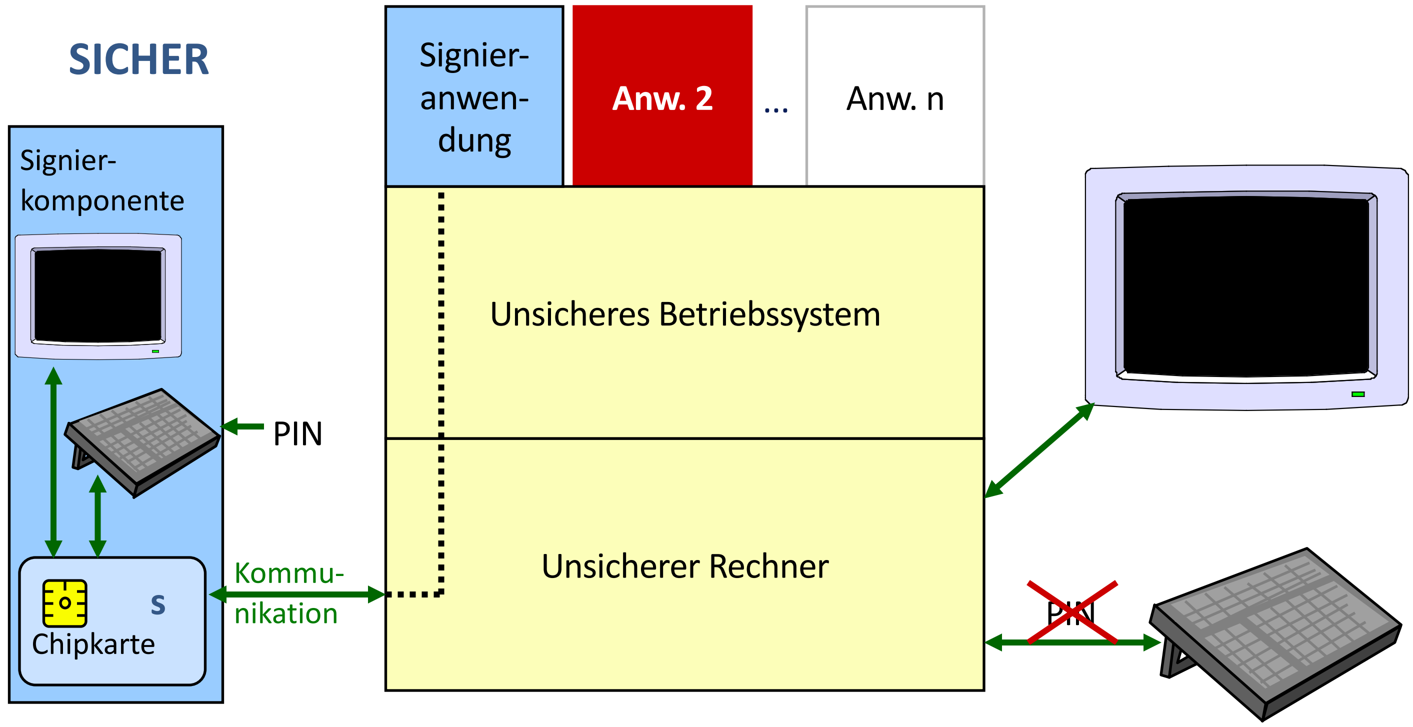
- Anzeige des Dokuments auf dem ext. Monitor
- Senden des Dokuments (bzw. dessen Hash-Wert) zur Chipkarte
- Aktivierung des Signiervorgangs auf der Karte durch PIN-Eingabe
- Rückgabe der Signatur an die Anwendung



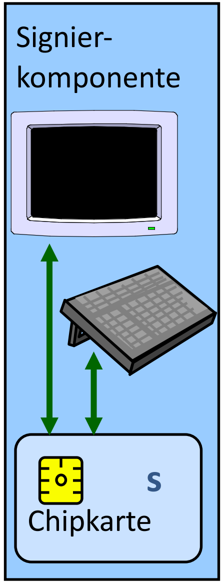
Standard-PC mit Chipkarte



Sichere Signierkomponente mit Standard-PC



Sichere Signierkomponente



=

- Display
- Tastatur
- Physischer Schutz: Manipulationserkennung
- Entwurf offengelegt (keine versteckten Trojanischen Pferde)

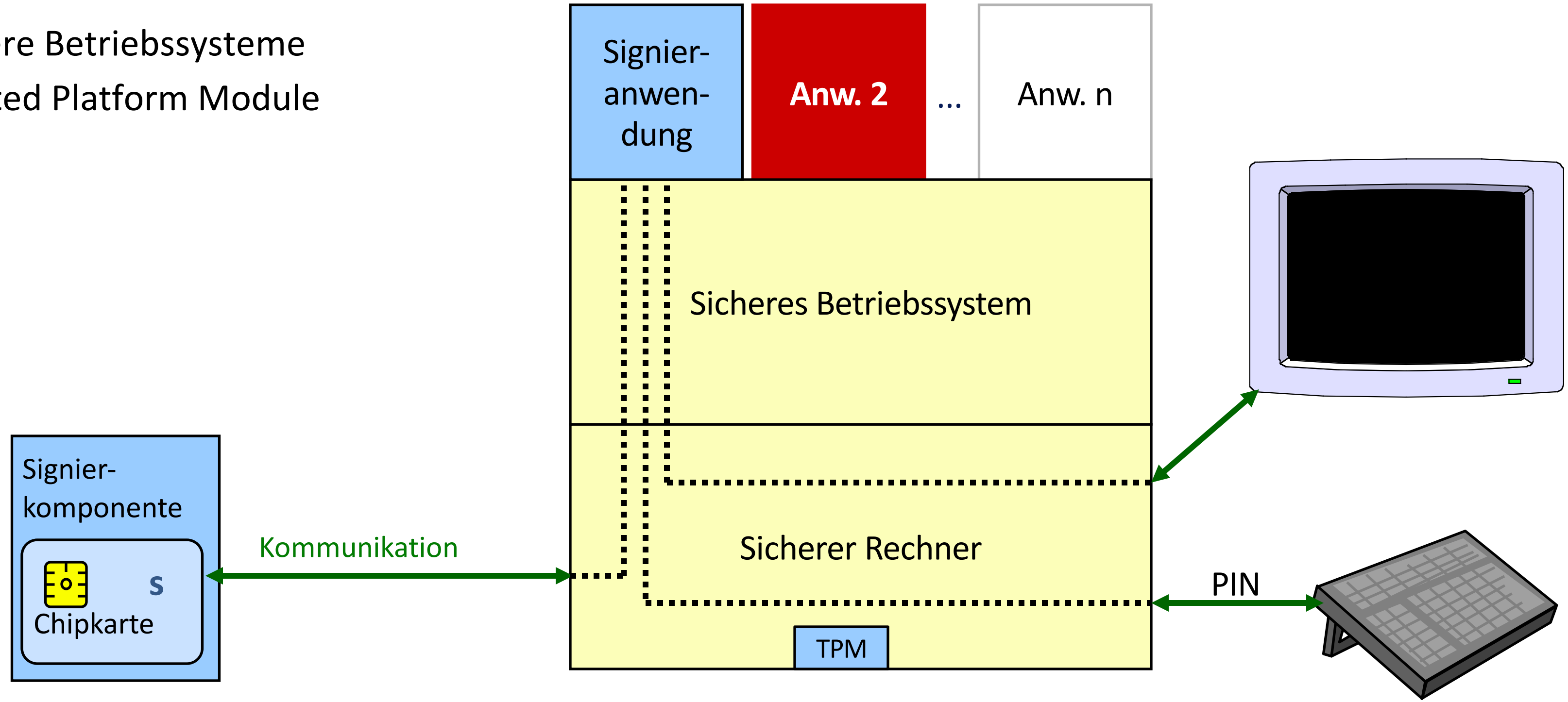
Chipkartenleser: Sicherheitsklassen

- **Klasse 1**
  - Keine Sicherheitsfunktionen
  - realisieren nur Kommunikation zwischen PC und Leser
- **Klasse 2**
  - PIN-Eingabe kann nicht vom PC mitgeloggt werden
    - Variante 1: PC-Tastatur ist direkt mit Leser verbunden, Verbindung zu PC wird während PIN-Eingabe (physisch) unterbrochen
    - Variante 2: Eigene Tastatur im Leser
- **Klasse 3**
  - eigene Tastatur und eigene Anzeige
  - PC ist nicht an der Kommunikation zwischen Karte, Tastatur und Anzeige beteiligt

- **Klasse 4**
  - eigener Signaturschlüssel
  - kann später ermittelt werden, in welchem Lesegerät die Signatur geleistet wurde

SICHER, wenn

- Physisch sichere Geräte
- sichere Betriebssysteme
- Trusted Platform Module



Detailaspekte zu PKI

Zeitstempeldienste und Selbstzertifizierung

Gültigkeitsmodelle für Zertifikate und Signaturen

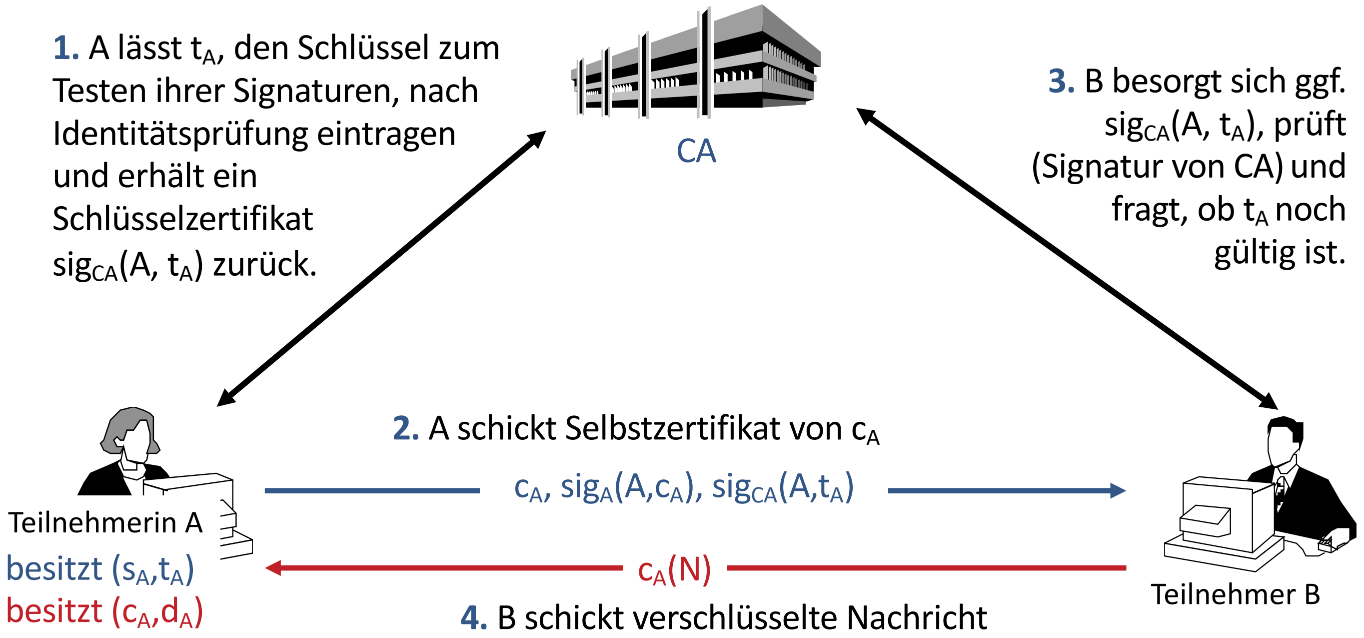
Zeitstempeldienste und Digitale Signatur

- Früher geleistete Signaturen können auch nach Kompromittierung des Signierschlüssels noch getestet werden.
- Frage: Wie verhindert man, dass rückwirkend gültige Signaturen erzeugt werden können? (Nach Kompromittierung des Signierschlüssels)

- Nachricht x erhält einen Zeitstempel (fest mit der Nachricht verbunden):
  - Teilnehmer S (Signierer)
    1. bildet Hash-Wert (Fingerabdruck) der Nachricht: h(x)
    2. fordert Zeitstempel von Zeitstempeldienst TS an: sig_{TS}(time,h(x))
    3. signiert Nachricht und Zeitstempel: sig_S(x, sig_{TS}(time,h(x)))
  - Solange der Zeitstempeldienst keine in der Vergangenheit (oder Zukunft) liegende Zeit signiert, kann eine bestimmte Nachricht nicht nachträglich signiert werden
  - Zeitstempel muss einen Fingerabdruck der Nachricht enthalten, damit nicht einfach frühere Zeitstempel der Form sig_{TS}(time) wiederverwendet werden können.

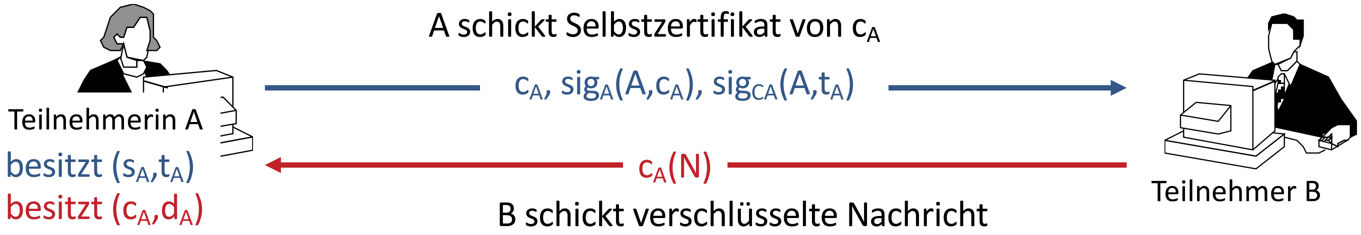
Selbstzertifizierung öffentlicher Verschlüsselungsschlüssel

Unter der Voraussetzung, dass eine funktionierende Infrastruktur für Digitale Signaturen existiert, wird keine zusätzliche für Verschlüsselung benötigt. Vorhandene Infrastruktur kann zur Sicherung der Authentizität der öffentlichen Verschlüsselungsschlüssel verwendet werden.



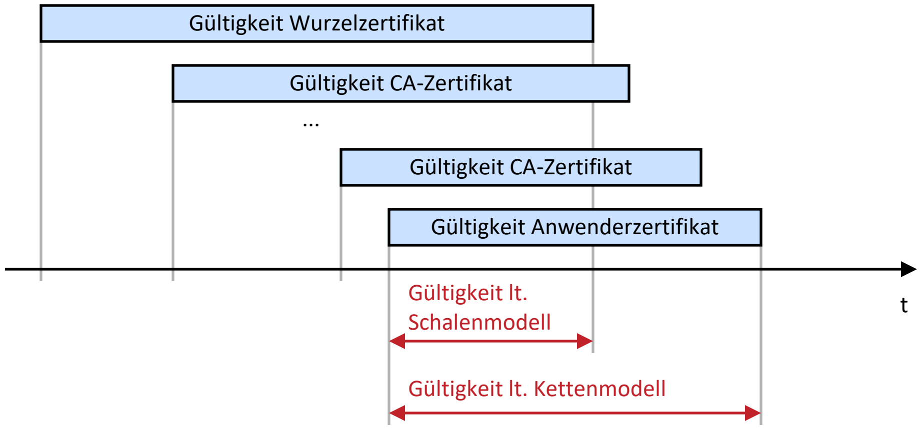
Selbstzertifizierung öffentlicher Verschlüsselungsschlüssel

- Trusted Third Parties werden gebraucht, wenn etwas bewiesen werden soll und nur dort.
  - Digitale Signatur: Beweisbarkeit erforderlich
  - Fremdzertifizierung des Testschlüssels durch Zertifizierungsstellen
- Bei asymmetrischer Verschlüsselung genügt es, sicher zu sein, dass der öffentliche Verschlüsselungsschlüssel authentisch ist.
  - Echtheit von Verschlüsselungsschlüsseln kann über vorhandene Infrastruktur für Digitale Signatur gesichert werden
  - Selbstzertifizierung des öffentlichen Verschlüsselungsschlüssels, jedoch keine Fremdzertifizierung



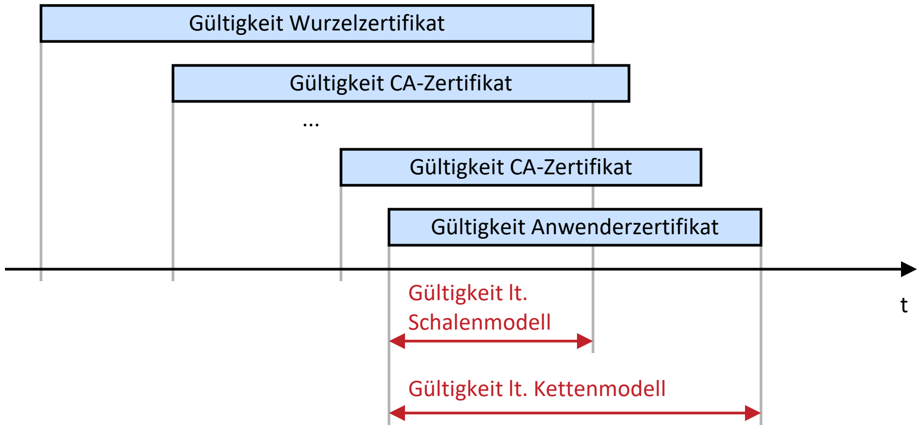
Gültigkeitsmodelle für Zertifikate und Signaturen

- Schalenmodell
  - Eine Signatur unter einem Dokument oder Zertifikat ist dann gültig, wenn zum Zeitpunkt ihrer Erstellung alle zugrunde liegenden Zertifikate gültig waren.
- Kettenmodell
  - Eine Signatur unter einem Dokument oder Zertifikat ist dann gültig, wenn zum Zeitpunkt ihrer Erstellung das zugehörige Schlüsselzertifikat gültig war.



Schalenmodell

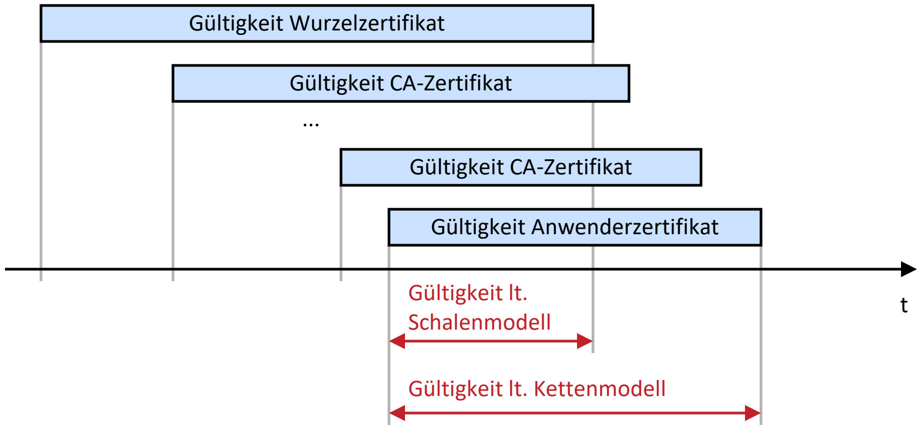
- Bei Ablauf oder Widerruf eines Zertifikats
  - Abgeleitete Zertifikate werden ungültig.
  - Beachte: Signaturen bleiben auch nach Ablauf der Zertifikate gültig, wenn Zertifikate zum Signierzeitpunkt gültig waren.
- Einfache praktische Umsetzung
  - Bei Überprüfung ist für alle Zertifikate derselbe Zeitpunkt maßgeblich.
  - int. Standard, basiert auf X.509 und RFC 3280 (PKIX-AG)



Kettenmodell

- Bei Ablauf oder Widerruf eines Zertifikats
  - Abgeleitete Zertifikate trotzdem bleiben gültig.
  - rekursive Anwendung auf alle zugrunde liegenden Zertifikate
- Basiert auf einer Forderung des § 19 (5) SigG:

  »Die Gültigkeit der von einem Zertifizierungsdiensteanbieter ausgestellten qualifizierten Zertifikate bleibt von der Untersagung des Betriebes und der Einstellung der Tätigkeit sowie der Rücknahme und dem Widerruf einer Akkreditierung unberührt.«



- Praxis
  - Übereinstimmende Gültigkeit von Schalenmodell und Kettenmodell bei folgender Situation