IT-Sicherheit: Signaturen & Zertifikate

Hash-Funktionen

Zentrale Funktionen für digitale Signaturen und digitale Zertifikate

Hash-Funktionen bilden beliebige Datenmengen auf Mengen einheitlicher Länge ab

Sie erzeugen den „digitalen Fingerabdruck“ einer Datei

Hash-Werte werden auch „Message Digest“ genannt

Ein Hash-Wert muss einer Datei eindeutig zugeordnet werden können

Beispiel für eine (sehr) einfache Hash-Funktion

Die Funktion bildet beliebige ganze Zahlen auf ihre einstellige Quersumme ab

56	→	5 + 6 =	11	→	1 + 1 =	**2**
132	→	1 + 3 + 2 =				**6**
8202	→	8 + 2 + 0 + 2 =	12	→	1 + 2 =	**3**
52572	→	5 + 2 + 5 + 7 + 2 =	21	→	2 + 1 =	**3**

Obacht! Das ist *keine* sinnvolle Funktion, sondern soll nur das Prinzip verdeutlichen …

Anforderungen an Hash-Funktionen

Hash-Funktionen müssen im IT-Sicherheits-Kontext (mindestens) zwei Bedingungen erfüllen:

Einwegfunktion

- Es darf nicht möglich sein, vom Hash-Wert auf den Ausgangswert zu schließen

Kollisionsresistenz

- In der Praxis müssen zwei verschiedene Ausgangswerte zwei verschiedene Hash-Werte erzeugen

Auch etwa **Lawineneffekt**: Kleine Veränderungen der Ausgangsmenge erzeugen maximale Veränderung im Hash-Wert

Erfüllt die simple Hash-Funktion die Anforderungen?

56	→	5 + 6 =	11	→ 1 + 1 =	**2**
132	→	1 + 3 + 2 =			**6**
8202	→	8 + 2 + 0 + 2 =	12	→ 1 + 2 =	**3**
52572	→	5 + 2 + 5 + 7 + 2 =	21	→ 2 + 1 =	**3**

Sie ist eine **Einwegfunktion** – man kommt vom Hash-Wert nicht zuverlässig zum Eingabewert

Sie ist offensichtlich **nicht kollisionsresistent**: Es gibt maximal 10 unterschiedliche Hash-Werte (0 bis 9), ab dem 11. Wert kommt es garantiert zu einer Kollision

Kollisionsresistenz

Jede Hashfunktion bildet *beliebig* große Datenmengen auf Mengen *einheitlicher* Länge ab

Die Menge der zu hashenden Daten ist potentiell *unendlich* – die der möglichen Hashwerte ist dagegen *endlich*

Es muss also irgendwann eine Kollision auftreten

Lösung: Möglichst viele Hashwerte ermöglichen

Bei einer Länge von 256 Bit können z.B. 2^{256} unterschiedliche Hashwerte gebildet werden (das ist eine 1 mit 77 Nullen …) – die Wahrscheinlichkeit einer Kollision wird extrem minimiert

Eine Hashfunktion muss *in der Praxis* kollisionsresistent sein – d.h.: solange keine Kollision nachgewiesen wurde, gilt sie als kryptographisch sicher.

Hash-Funktionen

Es gibt unzählige Hash-Funktionen, die bekanntesten:

- Die SHA-Gruppe (SHA-1, SHA-256, SHA-512 …)
- MD5
- RIPEMD-160
- Whirlpool

MD5 und SHA-1 ware lange Zeit Standard, es wurden mehrfach Kollisionen nachgewiesen: Veraltet und unsicher!

Für IT-Sicherheit ist vor allem **SHA-256** relevant

SHA-256: *Secure Hash Algorithm*, erzeugt 256 Bit große Hashwerte

D.h.: Jeder beliebig großen Datenmenge wird ein eindeutiger, 256 Bit großer Wert zugeordnet

Notation von Hash-Werten

Hash-Werte sind eine lange Folge von 1 und 0 – lassen sich nur unlesbar aufschreiben

Daher werden Hash-Werte üblicherweise als Hex-Wert notiert

Die 32-Bit-Zahl …

```
11111010 10110110 11010010 10111110
```

wird so zu

```
f    a    b    6    d    2    b    e
```

```
fab6d2be
```

Beispiele SHA-1

Hallo Welt

`28cbbc72d6a52617a7abbfff6756d04bbad0106a`

Hallo Welt!

`726c3e8861ab0652a5043ea5faff6d3ef33fb209`

Obwohl nur minimale Veränderungen vorliegen, werden komplett unterschiedliche Hash-Werte erzeugt

Hallo und guten Tag! Wie geht‘s?

`aedf6fce7d244cedb9e1a377aff3dd4da7e80585`

Ganz egal, wie groß die Datenmenge ist – der Hashwert ist immer gleich lang

Einsatz von Hashfunktionen

Immer, wenn es darum geht, zwei oder mehrere Dateien zu vergleichen, kommen Hashwerte ins Spiel

- **Datenübertragung:** An der Quelle und am Ziel werden Hashwerte der Datei(en) gebildet und verglichen. Sind sie gleich, war die Übertragung erfolgreich
- **Doubletten finden:** Hashwerte von Dateien lassen sich sehr viel schneller vergleichen als die Dateien selbst. Liegen doppelte Hashwerte vor, gibt es Doubletten
- **Passwörter speichern:** Passwörter werden nicht im Klartext, sondern als Hash gespeichert. Bei der Anmeldung wird das eingegebene Passwort gehasht und mit dem hinterlegten Hashwert verglichen

Hashfunktionen sind *keine* Verschlüsselung

Auch wenn ein Hashwert so wirkt, als würden da Daten „verschlüsselt“ – Hashfunktionen sind *keine* Verschlüsselung!

Verschlüsseln heißt: Unter Einsatz eines Schlüssels wird Klartext in ein Chiffrat überführt, das wieder in den Klartext umgewandelt werden kann. Und genau das geht bei Hashwerten nicht:

- Bei Hashfunktionen gibt es keinen „Schüssel“
- Hashfunktionen sind Einwegfunktionen – eine Überführung des Hashwerts in den Ausgangswert ist *nicht* möglich

Digitale Signaturen

Dateien werden nicht nur absichtlich oder unabsichtlich verändert, es kann auch technische Gründe geben (Übertragungsfehler etc.)

Die Manipulation einer Datei lässt sich also generell *nicht* verhindern

Mit einer digitalen Signatur lässt sich allerdings zuverlässig erkennen, ob eine Datei manipuliert bzw. verändert wurde

Schutzziel: Integrität.

Das Prinzip einer digitalen Signatur

Der Empfänger soll überprüfen können, ob eine Datei beim Transport verändert wurde

Lösungsansatz:

- Der Absender erzeugt einen Hash der Datei
- Der Absender verschickt die Datei zusammen mit dem Hashwert und nennt die benutzte Hashfunktion
- Der Empfänger hasht die Datei mit der angegebenen Funktion und erzeugt so seinerseits einen Hashwert
- Der Empfänger vergleicht die beiden Hashwerte. Sind sie identisch, wurde die Datei nicht manipuliert

Das funktioniert – hat aber einen großen Haken …

Das Problem

Die simple Übermittlung von Daten plus Hash ist sehr leicht angreifbar:

- Ein Angreifer fängt die Datei ab und manipuliert sie
- Der originale Hashwert wird verworfen
- Der Angreifer erzeugt einen Hashwert von der manipulierten Datei
- Der Angreifer schickt die manipulierte Datei zusammen mit dem neuen Hashwert an den Empfänger

Der Empfänger hat jetzt *keine* Möglichkeit, die Manipulation zu erkennen

Der originale Hashwert muss also besonders geschützt werden

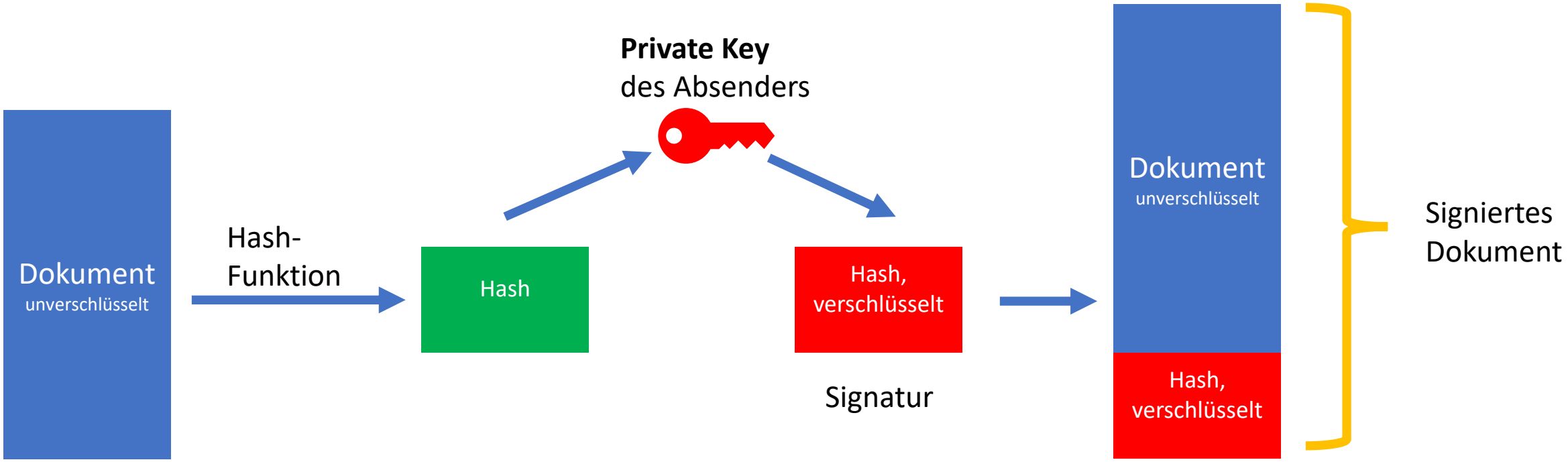
Hier kommt die asymmetrische Verschlüsselung ins Spiel

Asymmetrie lässt sich umkehren

Zur Erinnerung: Bei der asymmetrischen Verschlüsselung wird mit einem Schlüsselpaar aus Public & Private Key gearbeitet. Dabei gilt:

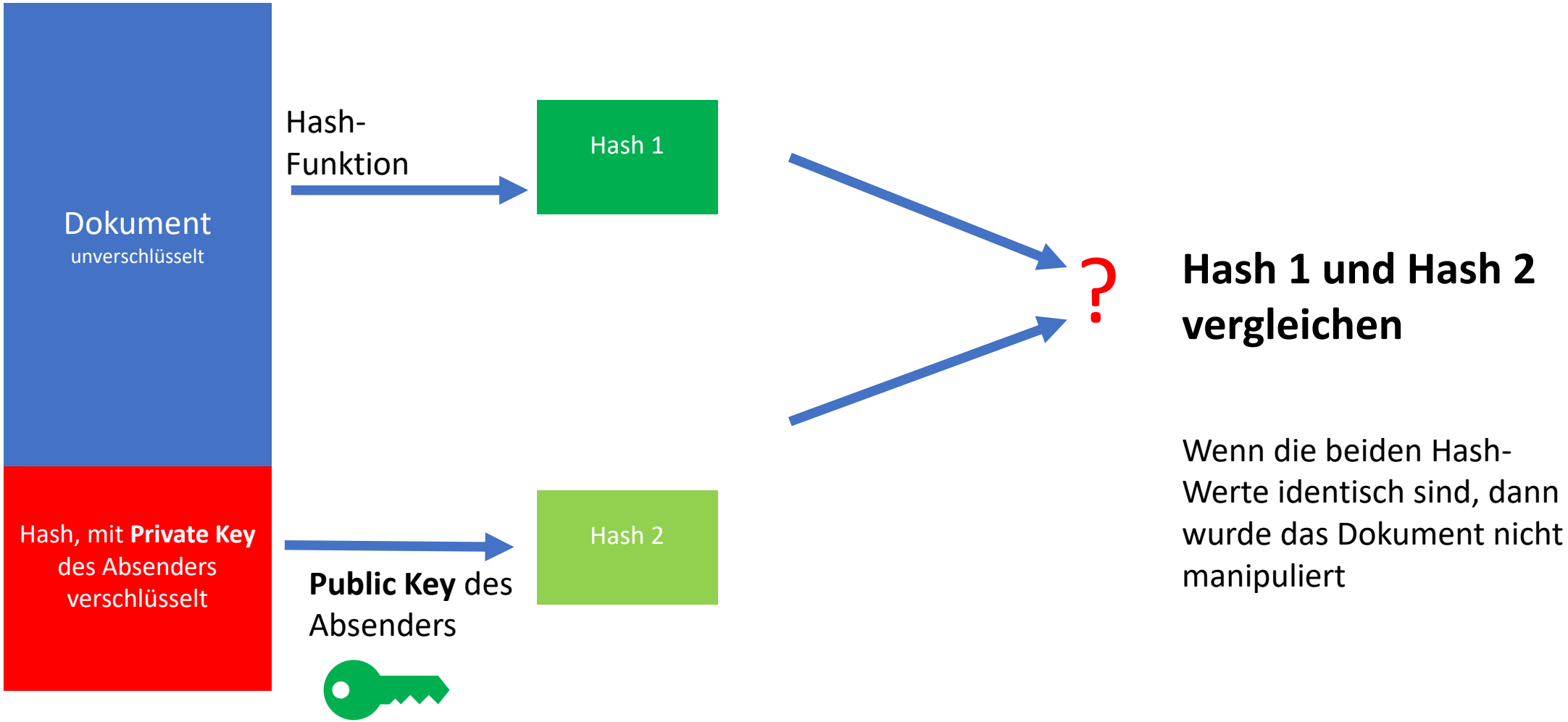
- Daten, die mit dem Public Key verschlüsselt wurden, lassen sich nur mit dem dazugehörigen Private Key wieder entschlüsseln (Schutzziel: Vertraulichkeit)
- Daten, die mit dem Private Key verschlüsselt wurden, lassen sich nur mit dem dazugehörigen Public Key wieder entschlüsseln (Schutzziel … garantiert *nicht* Vertraulichkeit ;-))

Signatur erstellen



Signaturfunktion: Die benutzte Hash-Funktion
Signaturschlüssel: Der Private Key des Signierenden
Signatur: Der verschlüsselte Hash-Wert

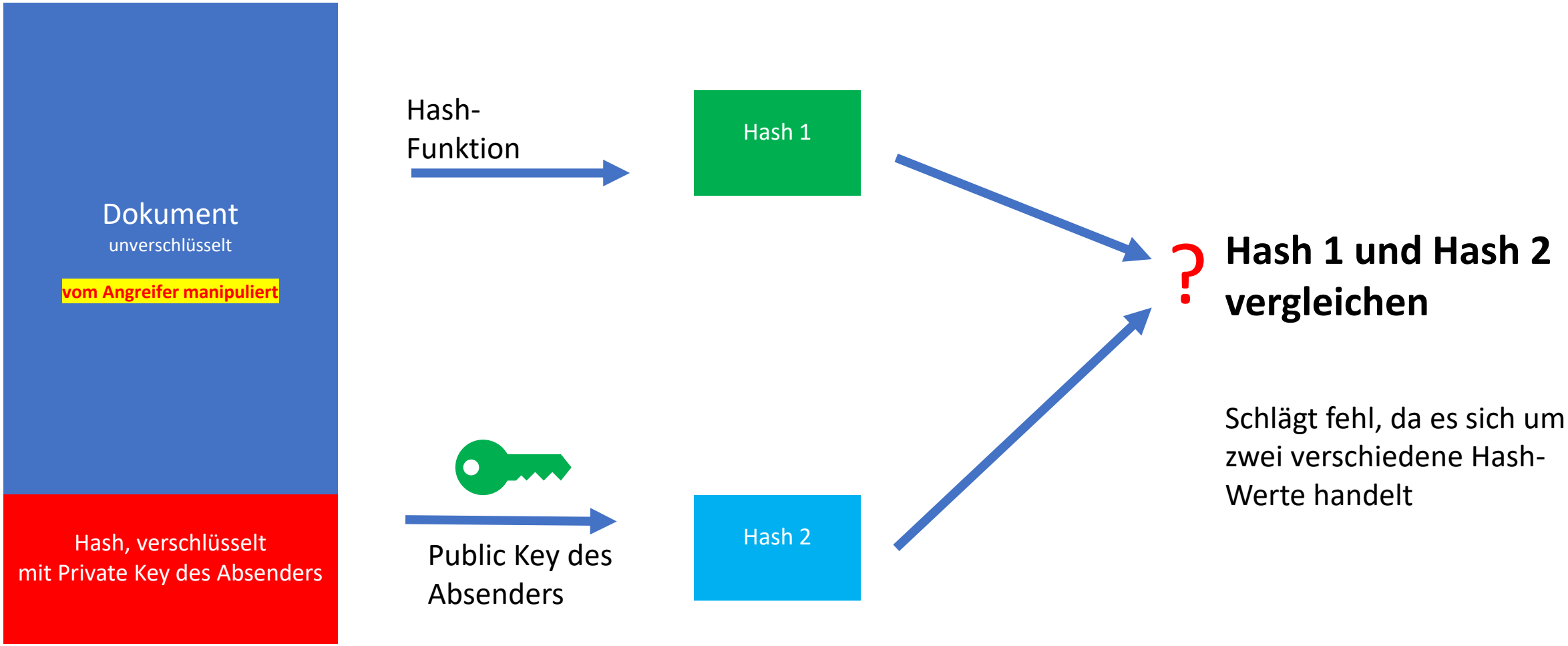
Signatur prüfen



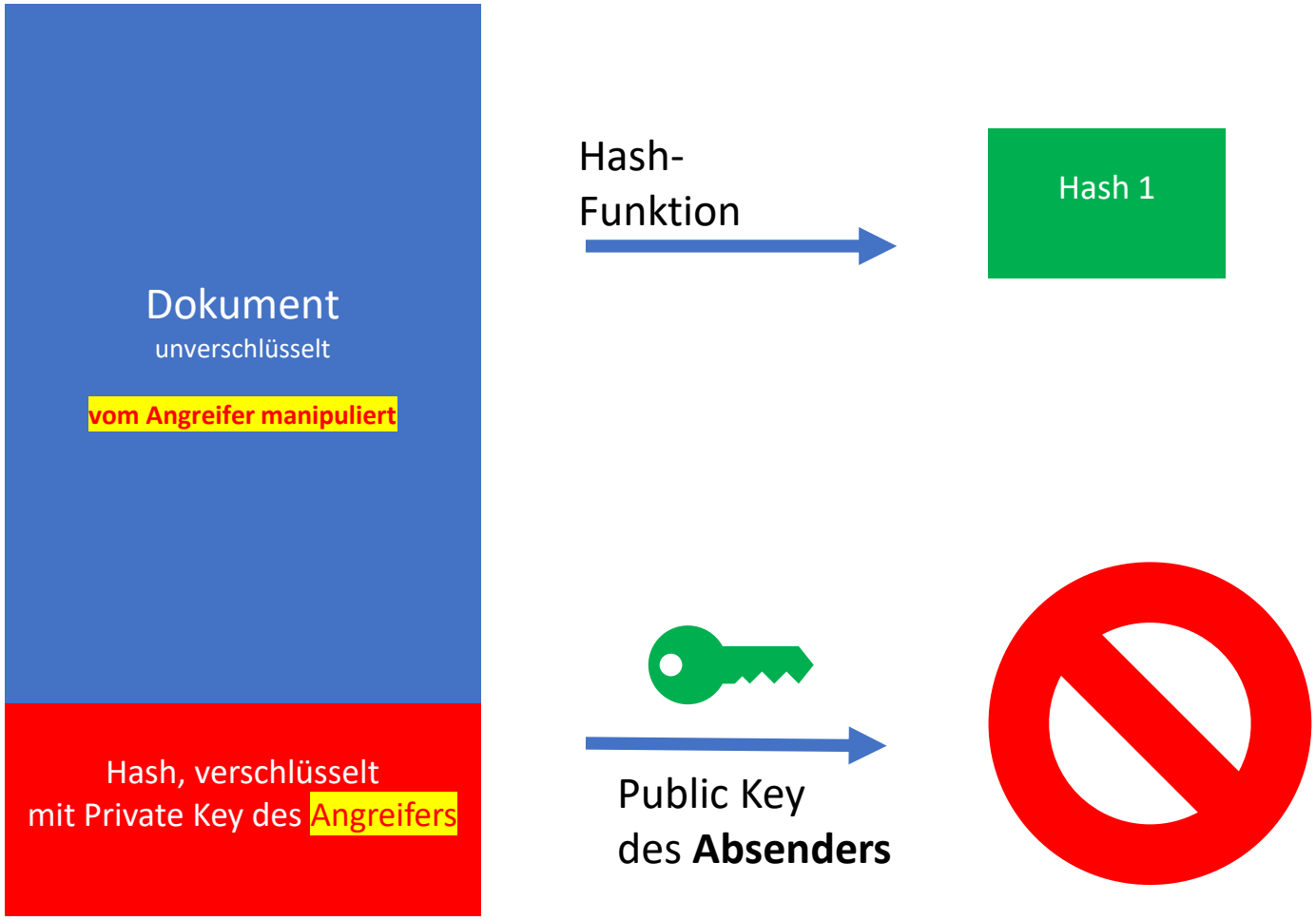
Hash 1 und Hash 2 vergleichen

Wenn die beiden Hash-Werte identisch sind, dann wurde das Dokument nicht manipuliert

Angriff: Das Dokument wurde manipuliert



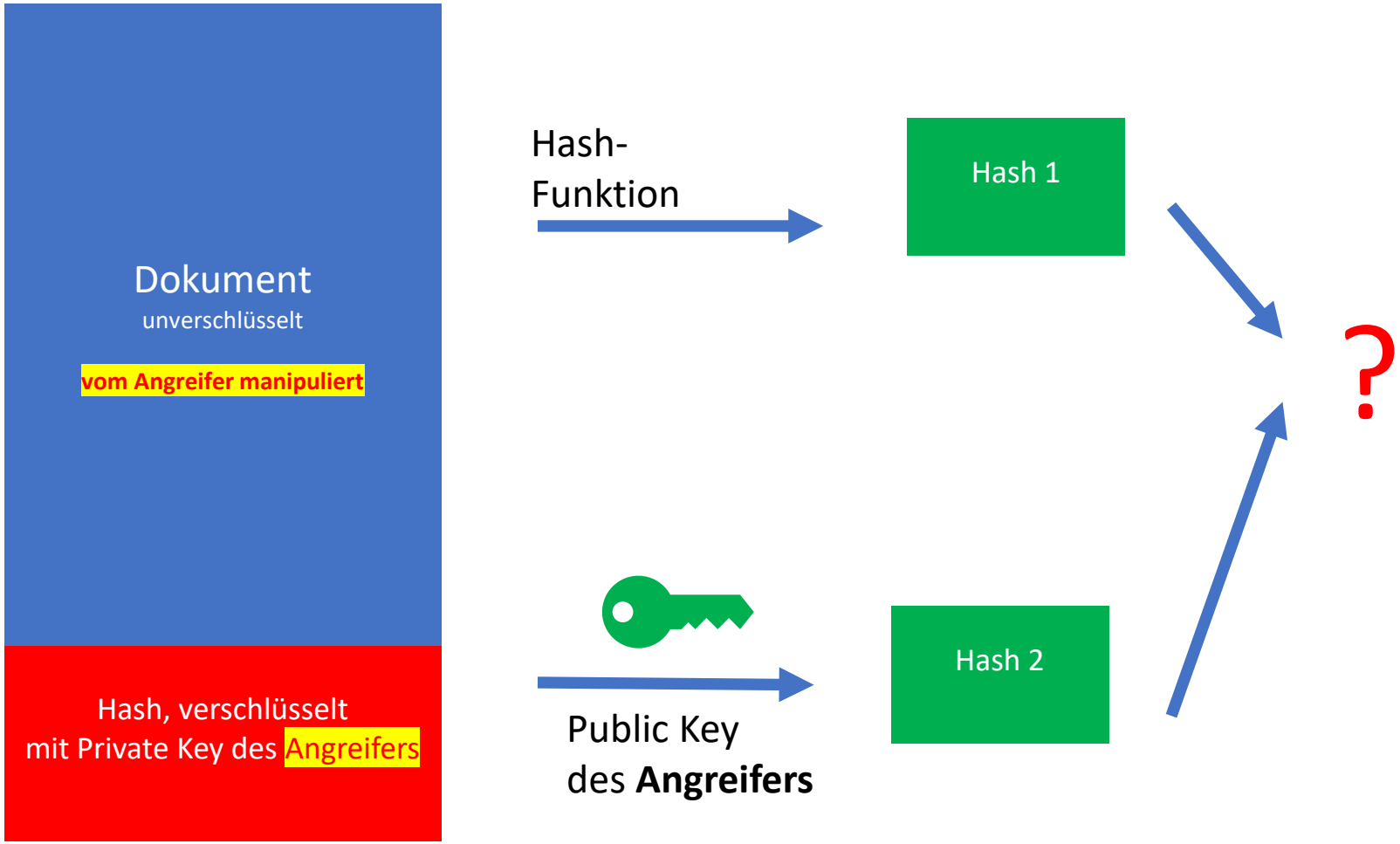
Angriff: Dokument wird manipuliert und vom Angreifer signiert



Schlägt fehl, da sich die Signatur nicht entschlüsseln lässt.

Die Signatur wurde mit dem Private Key des Angreifers erstellt und kann nur mit dem Public Key des Angreifers entschlüsselt werden.

Angriff: Manipuliert, vom Angreifer signiert, Angreifer hat uns seinen Public Key untergejubelt



Wir halten den Public Key des *Angreifers* für den Public Key des *Absenders*

Damit gelingt die Signaturprüfung und also auch der Angriff

Authentizität: Digitale Zertifikate

Zentrales Problem aller (asymmetrischen) Verschlüsselung: Wer garantiert die Echtheit eines Public Keys?

Wie können wir in der Praxis sicher sein, tatsächlich die Website der Bank, des Online-Händlers etc. erreicht zu haben?

Anders gesagt: Wie kann **Authentizität** gewährleistet werden?

Das Problem lässt sich mit kryptographischen Methoden allein *nicht* lösen, hier müssen wir jemanden vertrauen

Ein Zertifikat ist im Kern ein von einer „vertrauenswürdigen Stelle“ signierter Public Key des Zertifizierten

Problem: Zur Prüfung des signierten Public Key benötigt man den Public Key desjenigen, der da signiert hat – und damit sind wir wieder am Anfang: Wer garantiert die Echtheit dieses Public Key?

Bestandteile eines Zertifikats

Zertifikate enthalten zahlreiche Angaben, die im Standard X.509 definiert werden. Die wichtigsten:

- Name der Zertifizierungsstelle (wer hat zertifiziert?)
- Name des Zertifizierten (was wurde zertifiziert?)
- Die für die Signatur benutzte Hash-Funktion (wie wurde signiert?)
- Der Public Key des Zertifizierten (um den geht es ja)
- Gültigkeitsdauer des Zertifikats (normalerweise 1 oder 2 Jahre)
- Seriennummer des Zertifikats

Zertifikat: Die kombinierten Angaben (z.B. Name der Stelle, Name des Zertifizierten, Public Key, Seriennummer) werden von einer „vertrauenswürdigen Stelle“ signiert

Zertifikatsprüfung

Ein Zertifikat ist im Kern ein von einer vertrauenswürdigen Stelle signiertes Public Key

Eine Zertifikatsprüfung entspricht also einer Signaturprüfung:

- Die Signatur wird mit dem Public Key der vertrauenswürdigen Stelle entschlüsselt (= originaler Hashwert)
- Die Daten im Zertifikat werden gehasht
- Die beiden Hashwerte werden verglichen
- Sind sie identisch, ist das Zertifikat (und damit der Public Key) gültig

Die „vertrauenswürdigen Stellen“

Zur Prüfung eines Zertifikats (= Prüfung der Signatur) wird der Public Key der vertrauenswürdigen Stelle benötigt

Diese Public Keys dieser Stellen sind im System hinterlegt

Jedes Betriebssystem, auch jeder Browser, bringt von Haus aus eine Reihe von Public Keys mit, denen das System vertraut

Wer „vertrauenswürdig“ ist, entscheiden die Hersteller der benutzten Systeme

Wir können hier nur den Herstellern der Systeme / der Software vertrauen (Google, Microsoft, Apple …) – beurteilen können wir das nicht

PKI (Public Key Infrastructure)

Ein System zur Ausstellung, Verwaltung und Prüfung von Public Keys; besteht (vereinfacht) aus **RA**, **CA** und **VA**

- **RA:** Registration Authority: Registrierungsstelle. Nimmt Anträge auf ein Zertifikat entgegen und prüft sie. Nach erfolgreicher Prüfung reicht die RA den Antrag an die CA weiter.
- **CA:** Certificate Authority: Zertifizierungsstelle. Stellt das gewünschte Zertifikat aus und gibt es an den Antragsteller weiter. Informiert die VA über die Ausstellung des Zertifikats.
- **VA:** Validation Authority: Validierungsdienst. Überprüft auf Anfrage die Gültigkeit / Echtheit eines Zertifikats.

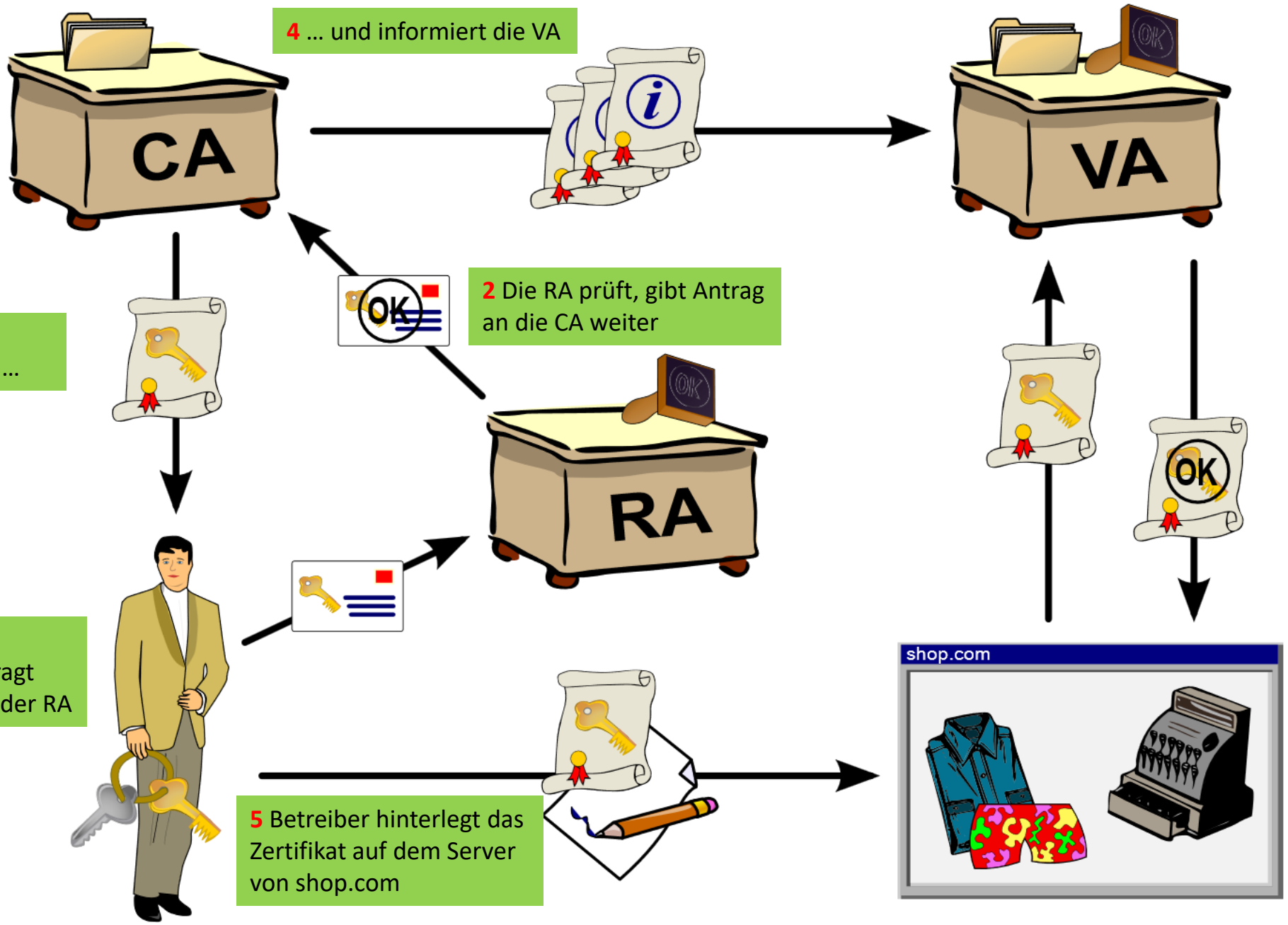


Bild:Wikipedia

Einfache elektronische Signatur

Sehr niedriges Sicherheitslevel, keine kryptographischen Methoden, kann leicht manipuliert werden

Keine genaue Definition, die Signatur kann z.B. aus einem Textbaustein am Ende einer Mail bestehen:

```
Rainer Zufall
Musterstr. 15
99999 Musterhausen
```

Auch: PDF unterschreiben, eingescannte Unterschrift, Unterschrift auf Lesegerät als Empfangsbestätigung etc.

Vor Gericht einsetzbar, hängt aber vom Gericht ab, ob sie akzeptiert wird, kann bestritten werden etc.

Fortgeschrittene Signatur

Hohes Sicherheitslevel, nur der Signierende ist in der Lage, eine valide Signatur zu erzeugen

Die Signatur basiert auf einem digitalen Zertifikat

Aussteller des Zertifikats spielt keine Rolle

Kann auch ein selbsterstelltes Zertifikat sein

Die Signatur ist eine Dateischnipsel als Anhang zu einem Dokument / einer Mail

Vor Gericht einsetzbar, Echtheit der Signatur muss aber u.U. bewiesen werden

Qualifizierte elektronische Signatur (QES)

Sehr hohes Sicherheitslevel, in allen EU-Staaten rechtlich gleichgestellt mit einer handschriftlichen Unterschrift

Technisch völlig identisch mit einer fortgeschrittenen Signatur, aber:

- Das zugrundeliegende Zertifikat wird erst nach Identitätsprüfung ausgestellt
- „qualifiziertes Zertifikat“ von anerkannten Zertifizierungsdiensten (also *keine* selbsterstellten Zertifikate)
- Die Signatur erfolgt über eine „sichere Signaturerstellungseinheit“ (SSEE; z.B. Kartenlesegerät mit spezieller Chipkarte für Signatur)
- Die SSEE muss von einer anerkannten Stelle geprüft worden sein
- Der Signaturschlüssel (= Private Key) darf nur in der SSEE gespeichert sein

Anerkannte Zertifizierungsdienste

Anerkannte Zertifizierungsdienste für Deutschland sind z.B.:

- D-Trust (Bundesdruckerei)
- Bundesnotarkammer
- Signtrust (Deutsche Post)
- S-Trust (Deutscher Sparkassen Verlag)
- Telesec (Deutsche Telekom)

Vollständige Liste aller europäischen Dienste:
https://eidas.ec.europa.eu/efda/trust-services/browse/eidas/tls