Kryptografie mit öffentlichen Schlüsseln

Kryptologie IL (Vorlesungsteil)

Andreas Unterweger

Studiengang ITS
FH Salzburg

Sommersemester 2024

Kurzwiederholung

- Begriffe
  - Klartext (weiter: Chosen-Plaintext-Angriff)
  - Chiffretext (weiter: Chosen-Ciphertext-Angriff)
  - Schlüssel
  - Verschlüsselung(-sfunktion)
  - Entschlüsselung(-sfunktion)
- Rechnerische Sicherheit
  - Brute-Force-Schlüsselsuche
  - Aktuelle Schlüssellängen
- Symmetrische vs. asymmetrische Kryptografie

Uberblick über Kryptografie mit öffentlichen Schlüsseln

- Asymmetrische Kryptografie
  - Ein Schlüssel ist öffentlich (z.B. zur Verschlüsselung)
  - Der zweite Schlüssel wird geheim gehalten (z.B. zur Entschlüsselung)
  - Begriff des Schlüsselpaares (engl. *public-private key pair*)
  - Es gibt **keinen geteilten geheimen Schlüssel**

  $\rightarrow$ Vereinfachter Schlüssel„tausch“
- Zur Erinnerung: Rechnerisch sichere Verfahren beruhen auf (Schwierigkeits-)Annahmen

- Beispiele:
  - Das Rivest-Shamir-Adleman-(RSA)-Kryptosystem
  - Kryptografie mit elliptischen Kurven (nur grober Ausblick)
- Fortgeschrittene Konzepte basierend auf asymmetrischer Kryptografie:
  - Digitale Signaturen
  - Public-Key-Infrastruktur

K.wh.: Modulare Arithmetik und multiplikative Gruppen I

- Modular Arithmetik
  - Die Modulo-Operation
  - Modul N
  - Kongruenz modulo N
  - Inverse und Umkehrbarkeit
- Multiplikative Gruppen
  - Was macht eine Gruppe aus?
  - Abgeschlossenheit
  - Existenz eines neutralen Elements
  - Umkehrbarkeit aller Gruppenelemente
  - Exponentiation
  - Generatoren
  - Teilgruppen
  - Diskreter Logarithmus

K.wh.: Modulare Arithmetik und multiplikative Gruppen II

- $\mathbb{Z}_n^* := \{1, 2, \cdots, n-1\}$ mit Multiplikation modulo $n \in \mathbb{P}$
  - Multiplikative Gruppe
  - Bisherige Einschränkung: n muss prim sein, damit alle Elemente umkehrbar sind
  - Generalisierung (ohne Beweis): Wenn $n \notin \mathbb{P}$, schließe alle Elemente aus, die nicht umkehrbar sind $\rightarrow$ Gruppen, die zusammengesetzte (nicht-prime) n erlauben

$\rightarrow$ Volle Definition: $\mathbb{Z}_N^* := \{a \in \{1, 2, \cdots, N-1\} | \operatorname{ggT}(a, N) = 1\}$ mit Multiplikation modulo $N \in \mathbb{N} \backslash \{0, 1\}$

- Beispiel $\mathbb{Z}_4^* = \{1, 3\}$ mit Multiplikation modulo 4
  - Abgeschlossenheit (folgt aus restlichen Bedingungen unten)
  - Existenz eines neutralen Elements: 1
  - Umkehrbarkeit: 1 (trivial), $3^{-1} \equiv 3 \pmod 4$, da $3 \cdot 3 \equiv 9 \equiv 1 \pmod 4$
  - 3 generiert $\mathbb{Z}_4^*$, da $3^0 := 1 \pmod 4$ und $3^1 \equiv 3 \pmod 4$
- Beispiel $\mathbb{Z}_8^* = \{1, 3, 5, 7\}$ mit Multiplikation modulo 8

Definition und Überblick I

- Definition von $\phi(N) := |\mathbb{Z}_N^*|$
  - Gibt an, wie viele Elemente in $\mathbb{Z}_N^*$ sind
  - Auch Eulersche Phi-Funktion genannt
  - Wenn $N \in \mathbb{P}$, ist $\phi(N) = N - 1$ (siehe vorherige Vorlesung)
  - Wenn $N \notin \mathbb{P}$, d.h., $N = \prod_i p_i^{e_i}$ unterschiedliche Primfaktoren p_i und Exponenten $e_i \in \mathbb{N}\backslash\{0\}$, ist $\phi(N) = \prod_i p_i^{e_i - 1} \cdot (p_i - 1)$; Beweis siehe Katz (2021)
- Beispiel: $\phi(4)$
  - Von vorher: $\phi(4) := |\mathbb{Z}_4^*| = |\{1, 3\}| = 2$
  - Alternativ: $\phi(4) = \phi\left(2^2\right) = 2^{2-1} \cdot (2 - 1) = 2^1 \cdot 1 = 2$
- Beispiel: $\phi(8)$
  - Von vorher: $\phi(8) := |\mathbb{Z}_8^*| = |\{1, 3, 5, 7\}| = 4$
  - Alternativ: $\phi(8) = \phi\left(2^3\right) = 2^{3-1} \cdot (2 - 1) = 2^2 \cdot 1 = 4$
- Beispiel: $\phi(15) = \phi\left(3^1 \cdot 5^1\right) = \left(3^0 \cdot (3 - 1)\right) \cdot \left(5^0 \cdot (5 - 1)\right) = (1 \cdot 2) \cdot (1 \cdot 4) = 2 \cdot 4 = 8$

Definition und Überblick II

→ $\phi(N)$ zu berechnen erfordert Wissen über die Faktorisierung von N

- Zur Erinnerung: Eine große Zahl zu faktorisieren wird als schwierig angenommen

→ RSA-Annahme

  - Wenn $\phi(N)$ oder die Faktorisierung von N bekannt ist, sind einige Operationen in $\mathbb{Z}_N^*$ leicht (mehr Details später)
  - Wenn $\phi(N)$ und die Faktorisierung von N nicht bekannt sind, sind diese Operationen in $\mathbb{Z}_N^*$ schwierig (angenommen es gibt keine Abkürzungen)

  → Wenn nur Bob $\phi(N)$ kennt, kann nur er Nachrichten entschlüsseln

- Überblick über RSA
  - Schlüsselerzeugung
  - Verschlüsselung
  - Entschlüsselung
  - Korrektheit
  - Praktische Belange

RSA-Schlüsselerzeugung

1. Generiere zwei unterschiedliche große Primzahlen p und q mit ungefähr gleicher Länge (Details in der Literatur; außerhalb des Umf.)
2. Berechne $N := p \cdot q$
   - Spätere Operationen werden in $\mathbb{Z}_N^*$ durchgeführt

   $\rightarrow$ $\phi(N) = (p-1) \cdot (q-1)$
   - N ist **nicht** geheim, aber seine Faktorisierung und $\phi(N)$ müssen **geheim** sein
3. Wähle ein $e \in \mathbb{Z}_N^*$, sodass $\mathrm{ggT}(e, \phi(N)) = 1$
   - e ist **öffentlich**

   $\rightarrow$ (e, N) ist der **öffentliche Schlüssel**
4. Berechne d als das Inverse von e, d.h., $e \cdot d \equiv 1 \pmod{\phi(N)}$
   - Algorithmen zur Berechnung sind außerhalb des Umfangs (Details in der Literatur)
   - d muss existieren, da e per Definition ein Inverses haben muss
   - d muss **geheim** bleiben

   $\rightarrow$ (d, N) ist der **geheime Schlüssel** (auch: private Schlüssel)

RSA-Verschlüsselung und -Entschlüsselung

- Klartext und Chiffretext sind Zahlen in $\mathbb{Z}_N^*$
- Verschlüsselung und Entschlüsselung sind Operationen in $\mathbb{Z}_N^*$
- $c := E(k_{oeffentlich}, m) = E((e, N), m) = m^e \pmod N$
- $m \stackrel{!}{=} D(k_{geheim}, c) = D((d, N), c) = c^d \pmod N$

- Beispiel mit kleinen Zahlen:
  1. Generiere $p = 11$ und $q = 13$
  2. Berechne $N = p \cdot q = 143$; $\phi(N) = (p-1) \cdot (q-1) = 10 \cdot 12 = 120$
  3. Wähle $e = 7$ (beachte, dass $\mathrm{ggT}(7, 120) = 1$)
  4. Berechne $d \equiv 103 \pmod{120}$
  5. Verschlüssle $m = 19$: $c := m^e \pmod N = 19^7 \pmod N \equiv 46 \pmod N$
  6. Entschlüssle $m \stackrel{!}{=} c^d \pmod N = 46^{103} \pmod N \equiv 19 \pmod N$

Korrektheit von RSA

- $m \stackrel{!}{=} D(k_{geheim}, E(k_{oeffentlich}, m)) = D((d, N), E((e, N), m)) = D((d, N), m^e \pmod N) = (m^e)^d \pmod N = m^{e \cdot d} \pmod N$

$\rightarrow$ $m \stackrel{!}{\equiv} m^{e \cdot d} \pmod N$

$\rightarrow$ $m^{e \cdot d} \stackrel{!}{\equiv} m^1 \pmod N$

- m generiert zyklisch eine Teilgruppe von $\mathbb{Z}_N^*$ (wiederholt sich nach der Teilgruppengröße)
- $\mathbb{Z}_N^*$ hat $\phi(N)$ Elemente
- Alle Teilgruppen von $\mathbb{Z}_N^*$ haben eine Größe, die ein ganzzahliger Teiler von $\phi(N)$ ist (Details in Katz (2021)), z.B. 60 Elemente für $m = 19$ in $\mathbb{Z}_{143}^*$

$\rightarrow$ $m^{\phi(N)} \equiv m^1 \pmod N$

- Da $e \cdot d \equiv 1 \pmod{\phi(N)} \rightarrow m^{e \cdot d} \equiv m^1 \pmod N$

Praktische Belange I

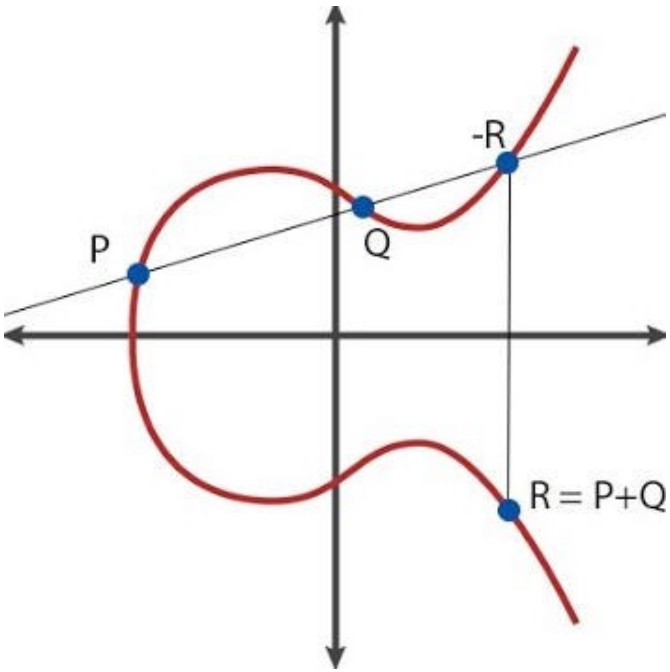
- „RSA aus dem Lehrbuch“ ist **nicht** sicher
  - Kleine Nachrichten können Angriffe vereinfachen
  - $\rightarrow$ Auffüll-Bits verwenden, um Struktur aus der Nachricht zu entfernen
  - RSA mit üblicherweise verwendetem Padding ist sicher gegenüber Chosen-Plaintext-, aber nicht Chosen-Ciphertext-Angriffen (ohne Details)
- RSA kann Zahlen größer als $N-1$ in $\mathbb{Z}_N^*$ nicht verschlüsseln
  - Möglichkeit 1: Zahl in Teile aufspalten (ähnlich wie bei Blockchiffren) und einen zufälligen Schlüssel für die tatsächliche Verschlüsselung aller separaten (aufgefüllten) Teile verwenden
  - Möglichkeit 2: Hybride Verschlüsselung (später) – verschlüssle oder leite (aufgefüllten) symmetrischen Schlüssel ab und setze Kommunikation mit symmetrischer Chiffre fort
- RSA kann Zahlen außerhalb von $\mathbb{Z}_N^*$ nicht verschlüsseln
  - Binärstring (Nachricht) in Zahl umwandeln (mit Auffüllen!)
  - Theoretischer Randfall: $m \notin \mathbb{Z}_N^*$, z.B., $11 \notin \mathbb{Z}_{143}^*$ (Korrektheit gilt dennoch und Angriffe bleiben schwierig; ohne Details)

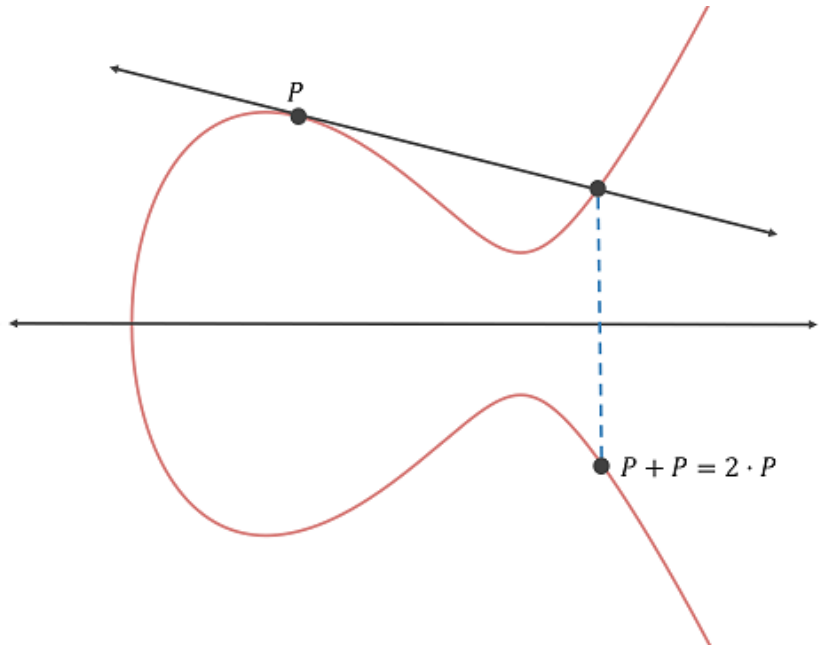
Praktische Belange II

- RSA ist relativ aufwändig zu berechnen
  - RSA nur für kurze Nachrichten verwenden, z.B. Schlüsseltausch
  - Symmetrische Chiffren für längerer Nachrichten verwenden
- Wie eine sichere Größe für das Faktorisierungsmodul N wählen
  - Asymmetrische Schlüsselgrößen **können nicht** mit symmetrischen Schlüsselgrößen verglichen werden
  - Asymmetrische Schlüsselgrößen sind nicht mit (Teil-)Gruppengrößen vergleichbar
  - Empfehlungen unter `https://www.keylength.com/en/compare/`
- Wahl der Werte
  - p und q: Müssen ähnlich groß sein (ansonsten Vorteil für einen Angreifer)
  - e: Kann klein sein, z.B. üblicherweise 3 oder 65537
  - d: Abgeraten, obwohl es gewählt und e als sein Inverses berechnet werden könnte; darf nicht zu klein sein (ansonsten Vorteil für einen Angreifer)

Ausblick: Kryptografie mit elliptischen Kurven I

- Elliptische Kurven
  - Spezielle Kurven (ohne Details), deren Punkte eine Gruppe bilden
  - Spezielle Definition von Addition: Schnittgeraden von Punkten mit Kurve mit Spezialfällen im Unendlichen (ohne Details)

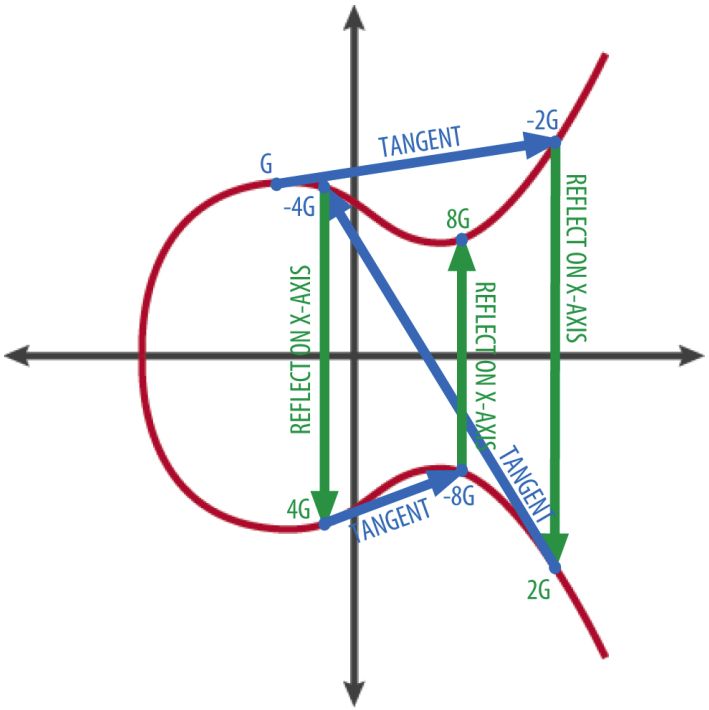


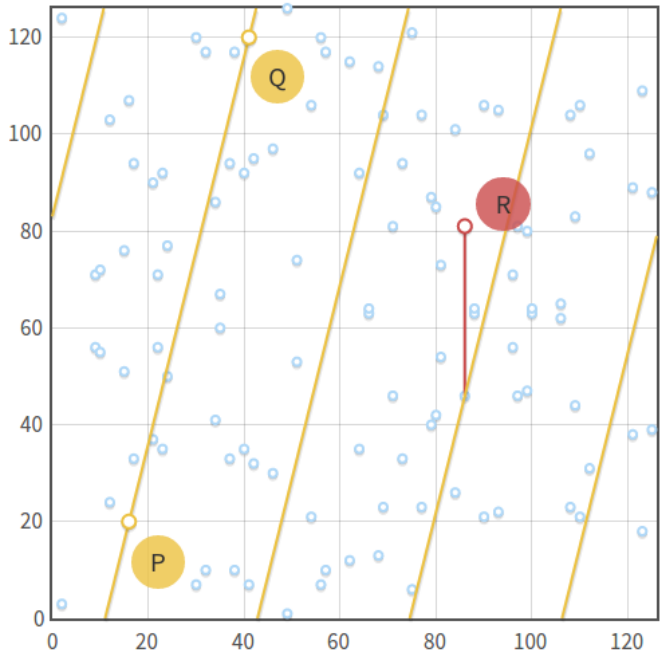


Quellen: Reid, P.: Part 2 – Is Elliptic Curve Cryptography (ECC) a step towards something more – Understanding ECC. https://medium.com/@peterreid_12788/part-2-is-elliptic-curve-cryptography-ecc-a-step-towards-something-more-understanding-ecc-3c933d3922e (Zugriff am 27.9.2022), 2016; Yang, H.: Same Point Addition on an Elliptic Curve. http://www.herongyang.com/EC-Cryptography/Introduction-Same-Point-Addition-on-Elliptic-Curve.html (Zugriff am 27.9.2022), 2022.

Ausblick: Kryptografie mit elliptischen Kurven II

- Multiplikation als wiederholte Addition eines Startpunktes definiert
- Kurvengleichung und Punktkoordinaten eigentlich in Primmodul





Quellen: Uszak, P.: How does ECC go from decimals to integers? https://crypto.stackexchange.com/q/48657 (Zugriff am 27.9.2022), 2017; Corbellini, A.: Elliptic Curve Cryptography: finite fields and discrete logarithms. https://andrea.corbellini.name/2015/05/23/elliptic-curve-cryptography-finite-fields-and-discrete-logarithms/ (Zugriff am 13.9.2022), 2015.

[4] Corbellini, A.: Elliptic Curve Cryptography: finite fields and discrete logarithms. https://andrea.corbellini.name/2015/05/23/elliptic-curve-cryptography-finite-fields-and-discrete-logarithms/ (Zugriff am 13.9.2022), 2015.

Ausblick: Kryptografie mit elliptischen Kurven III

- (Teilgruppenbildende) Generatoren können basierend auf Multiplikation analog zu multiplikativen Gruppen definiert werden (ohne Details)
- Der diskrete Logarithmus (basierend auf Multiplikation) in Elliptischen-Kurven-Gruppen wird als besonders schwierig zu berechnen angenommen
  - → Als Basis für asymmetrisches Kryptosystem verwenden (ohne Details)
  - → Benötigt kleinere Schlüssellängen im Vergleich zu Kryptosystemen, die auf dem Diskreten-Logarithmus-Problem basierend auf Exponentiation aufbauen (siehe Empfehlungen unter `https://www.keylength.com/en/compare/`)

[4] Corbellini, A.: Elliptic Curve Cryptography: finite fields and discrete logarithms. `https://andrea.corbellini.name/2015/05/23/elliptic-curve-cryptography-finite-fields-and-discrete-logarithms/` (Zugriff am 13.9.2022), 2015.

Überblick über digitale Signaturen

- Vom Unterzeichner/Sender generiert und an die Nachricht angehängt
- Kann vom Empfänger (oder jedem anderen) überprüft werden
- Bewahrt die Nachrichtenintegrität anstatt der -privatsphäre
- Garantien
  - Die signierte Nachricht wurde nicht verändert
  - Die signierte Nachricht stammt vom Unterzeichner

  $\rightarrow$ Nichtabstreitbarkeit: Der Unterzeichner kann nicht behaupten, die Nachricht **nicht** unterzeichnet zu haben
- Anwendungsbeispiel: Softwareupdate-/-patchüberprüfung
  - Alice (Anbieter) veröffentlicht ein Update mit einer Signatur
  - Bob (Benutzer) kann überprüfen, ob die Signatur korrekt ist
  - Wenn die Signatur inkorrekt ist, z.B. durch eine bösartige Veränderung, kann Bob das Update ablehnen
- Signaturen basieren auf Kryptografie mit öffentlichen Schlüsseln und Hashes

Kurzwh.: Asymmetrische Kryptosysteme und Hashes

- Asymmetrische Kryptografie
  - Ein Schlüssel ist öffentlich (z.B. zur Verschlüsselung)
  - Der zweite Schlüssel wird geheim gehalten (z.B. zur Entschlüsselung)
  - Notation: $c = E(k_{geheim}, m)$ und $m \stackrel{!}{=} D(k_{oeffentlich}, c)$
- Kryptografische Hashes
  - erzeugen einen Fingerabdruck/Hash einer gegebenen Nachricht
  - ordnen einer variabel langen Eingabe eine Ausgabe fester Länge zu
  - sind eine „Einbahn“, d.h. rechnerisch unzumutbar sie umzukehren
  - schützen Integrität (können verwendet werden, um Verfälschungen/Sabotage zu erkennen)
  - Notation: $h = H(m)$

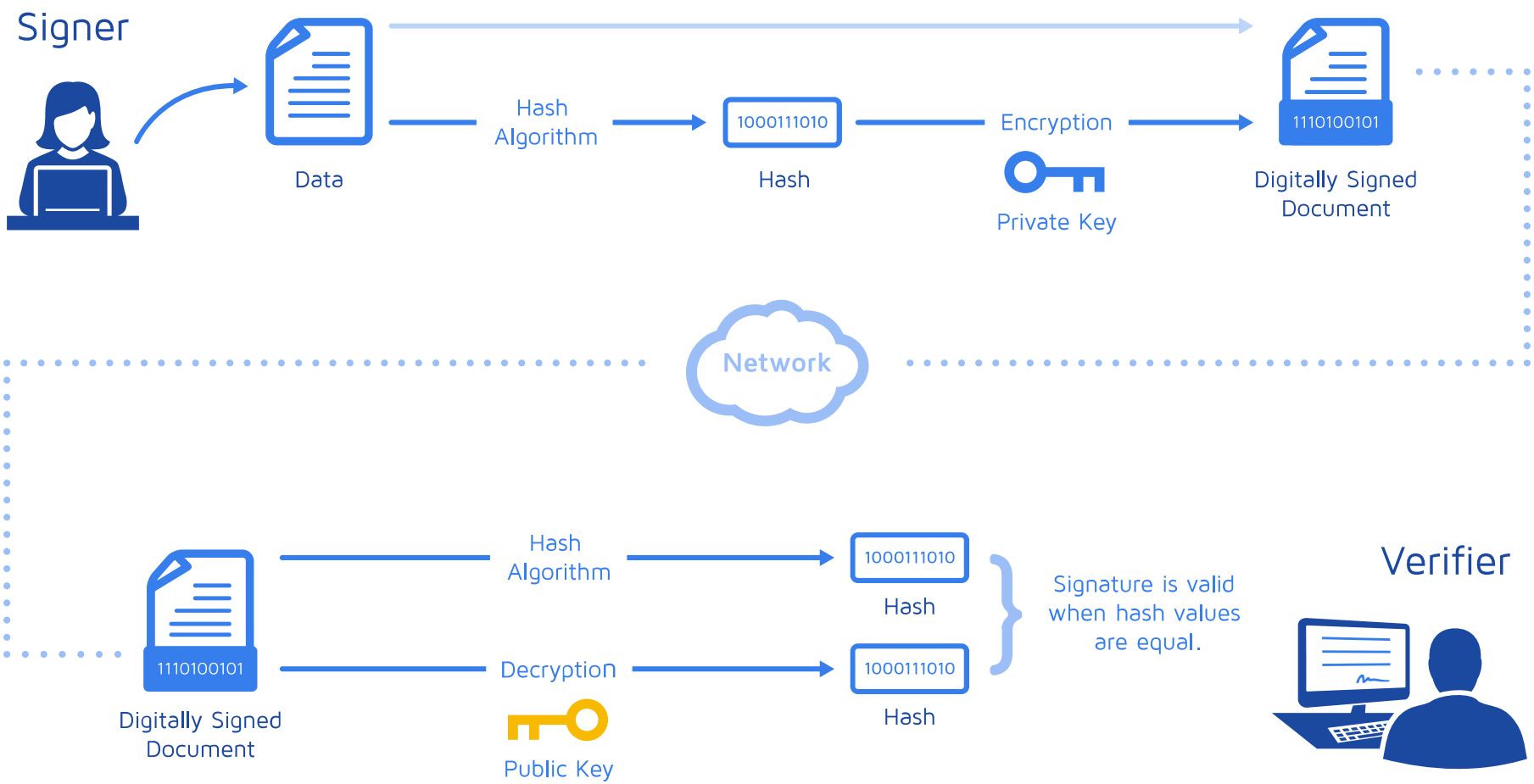
Signieren

- Arbeitet mit Hashes der Originalnachricht
  - Geringerer Rechenaufwand für längere Nachrichten erforderlich
  - Signatur hat konstante Länge
  - Sicherheit beruht auf Sicherheit des asymmetrischen Kryptosystems **und** der Sicherheit der Hashfunktion
- Notation: $s = S(k_{geheim}, m) := E(k_{geheim}, H(m))$
- Schritte des Signierens
  1. Alice generiert ein Schlüsselpaar
  2. Alice berechnet den Hash der zu sendenden Nachricht
  3. Alice verschlüsselt den Hash, um die Signatur zu erhalten
  4. Alice sendet die Signatur zusammen mit der Nachricht an Bob

Signaturüberprüfung I

- Überprüfungsschritte (angenommen Bob kennt Alices öffentlichen Schlüssel)
  1. Bob empfängt die Nachricht m' und ihre Signatur s
  2. Bob berechnet den Hash $h' = H(m')$ der Nachricht m'
  3. Bob entschlüsselt den Hash aus der Signatur: $h = D(k_{oeffentlich}, s)$
  4. Bob vergleicht den Hash h', den er berechnet hat, mit dem Hash h aus der Signatur
- Eine Signatur ist **nur** gültig, wenn $h = h'$
- Wenn $h = h'$, wurde die Nachricht nicht verändert **und** ist von Alice
- Wenn $h \neq h'$, ist **entweder**
  - die Nachricht verfälscht worden (und hat so ihren Hash geändert)
  - die Signatur nicht von Alice (sondern einem anderen Schlüsselpaar)

Signaturüberprüfung II



Quelle: DocuSign, Inc.: Understanding digital signatures. `https://www.docusign.com/how-it-works/electronic-signature/digital-signature/digital-signature-faq` (Zugriff am 13.9.2022), 2022.

Praktische Belange

- Zur Erinnerung: Die Sicherheit einer Signatur beruht auf der Sicherheit des asymmetrischen Kryptosystems **und** der Sicherheit der Hashfunktion
  - → Verwende sichere Kryptosysteme und sichere Hashfunktionen
    - Verwende ausreichende Schlüssel- bzw. Ausgabelängen
- Stufen digitaler Signaturen, z.B. europäische e-Signaturen
  - Einfach: „Herkömmliche" digitale Signatur
  - Fortgeschritten: Identifiziert zusätzlich den Unterzeichner (und mehr) und ist an diesen gebunden
  - Qualifiziert: Zusätzlich von speziellem Signaturgerät erzeugt (und mehr)
- Identitätsproblem: Wie Alice mit ihrem öffentlichen Schlüssel in Verbindung bringen?

→ Public-Key-Infrastruktur

[1] European Commission: eSignature FAQ.
`https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/eSignature+FAQ` (Zugriff am 13.9.2022), 2022.

Überblick über Public-Key-Infrastruktur

- Public-Key-Infrastruktur (PKI)
  - Architektur, die es erlaubt, öffentliche Schlüssel mit ihren „Besitzern“ zu assoziieren
  - Löst Schlüsselverwaltung und -verteilung
  - Erfordert Vertrauen in eine oder mehrere Parteien
- Unterschiedliche Modelle für PKI; Schwerpunkt auf dem/den Weitverbreitetsten

- Schwerpunktthemen:
  - Digitale Zertifikate
  - Zertifizierungsstellen und -hierarchien
  - Alternative Infrastrukturkonzepte

Digitale Zertifikate I

- Zur Erinnerung: Digital Signaturen erlauben es nur, Unterzeichner an Hand ihres öffentlichen Schlüssels zu überprüfen (sie überprüfen nicht, ob der Schlüssel einer Person oder Entität gehört)

$\rightarrow$ Verbindung zwischen Person/Entität und ihrem öffentlichen Schlüssel benötigt

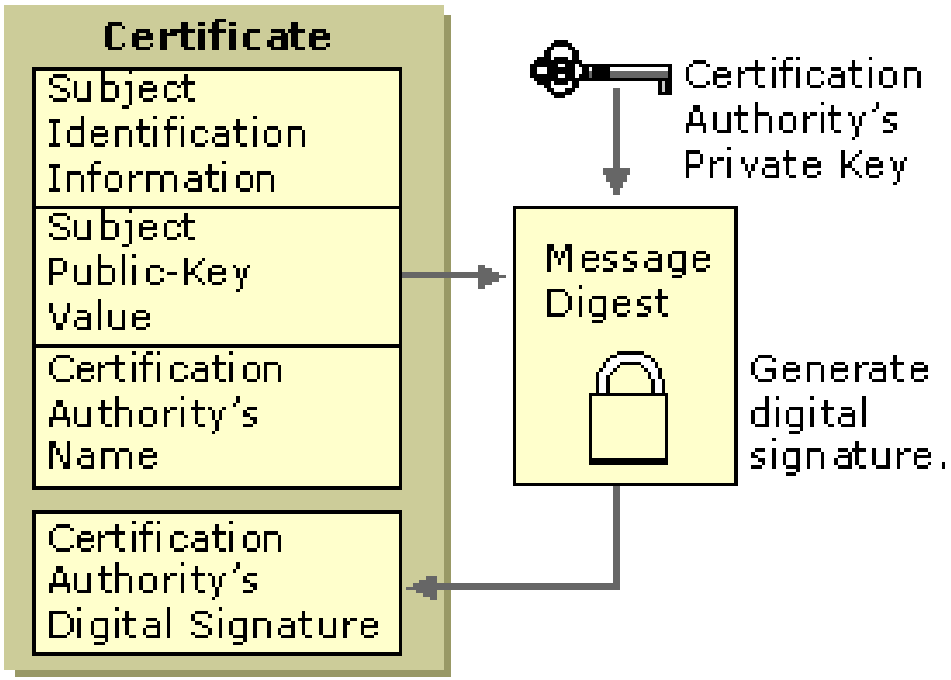
$\rightarrow$ Digitales Zertifikat: Assoziiert öffentlichen Schlüssel mit Person/Entität

  - Annahme: Vertrauenswürdige Partei T, deren öffentlicher Schlüssel allen Parteien bekannt ist
  - T signiert eine Aussage wie „Alices öffentlicher Schlüssel ist $k^A_{oeffentlich}$“
  - Bob kann mit dem öffentlichen Schlüssel von T die Aussage überprüfen
  - Notation: $C_{A\rightarrow T}$: T zertifiziert As Schlüssel (mit seinem eigenen geheimen Schlüssel)

- Notation: $C_{A\rightarrow T}$: As Schlüssel ist durch T zertifiziert
- $C_{A\rightarrow T} := S(k^T_{geheim}, \text{„Alices öffentlicher Schlüssel ist } k^A_{oeffentlich}\text{“})$
- Zertifikat kann durch seine Signatur von jedem überprüft werden

Digitale Zertifikate II

- Praktisch signierte Nachricht in digitalem Zertifikat: Separate Felder für Schlüssel und identifizierende Information mit kombiniertem Hash
- Beispielzertifikatsstandard: X.509

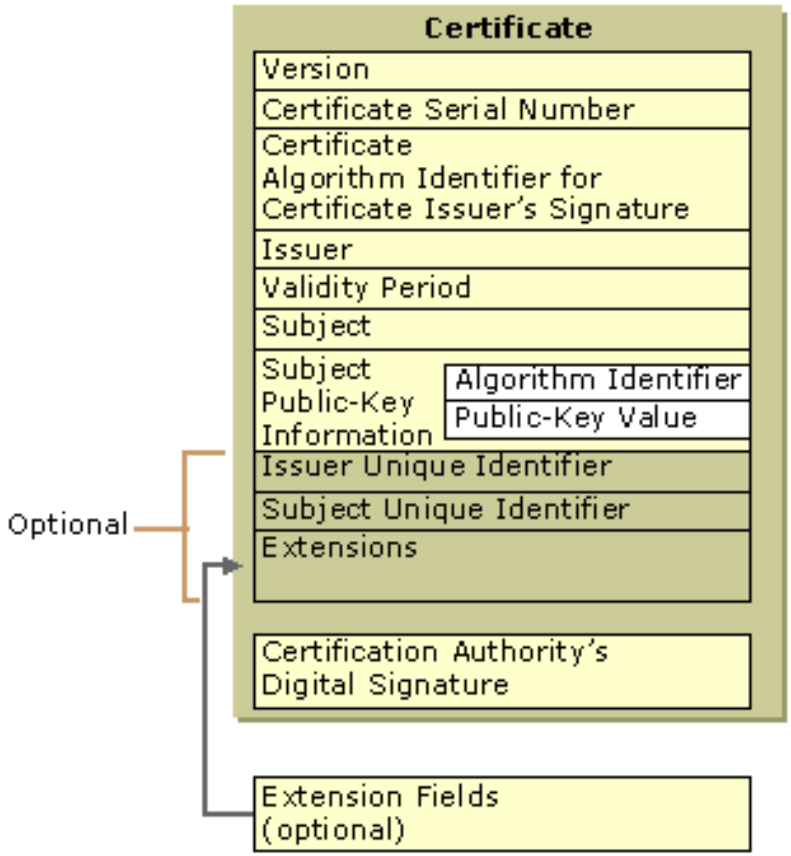


Quelle: Microsoft: Digital Certificates. https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/cc962029(v=technet.10)?redirectedfrom=MSDN (Zugriff am 13.9.2022), 2012.

[2] Microsoft: Digital Certificates. https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/cc962029(v=technet.10)?redirectedfrom=MSDN (Zugriff am 13.9.2022), 2012.

Digitale Zertifikate III

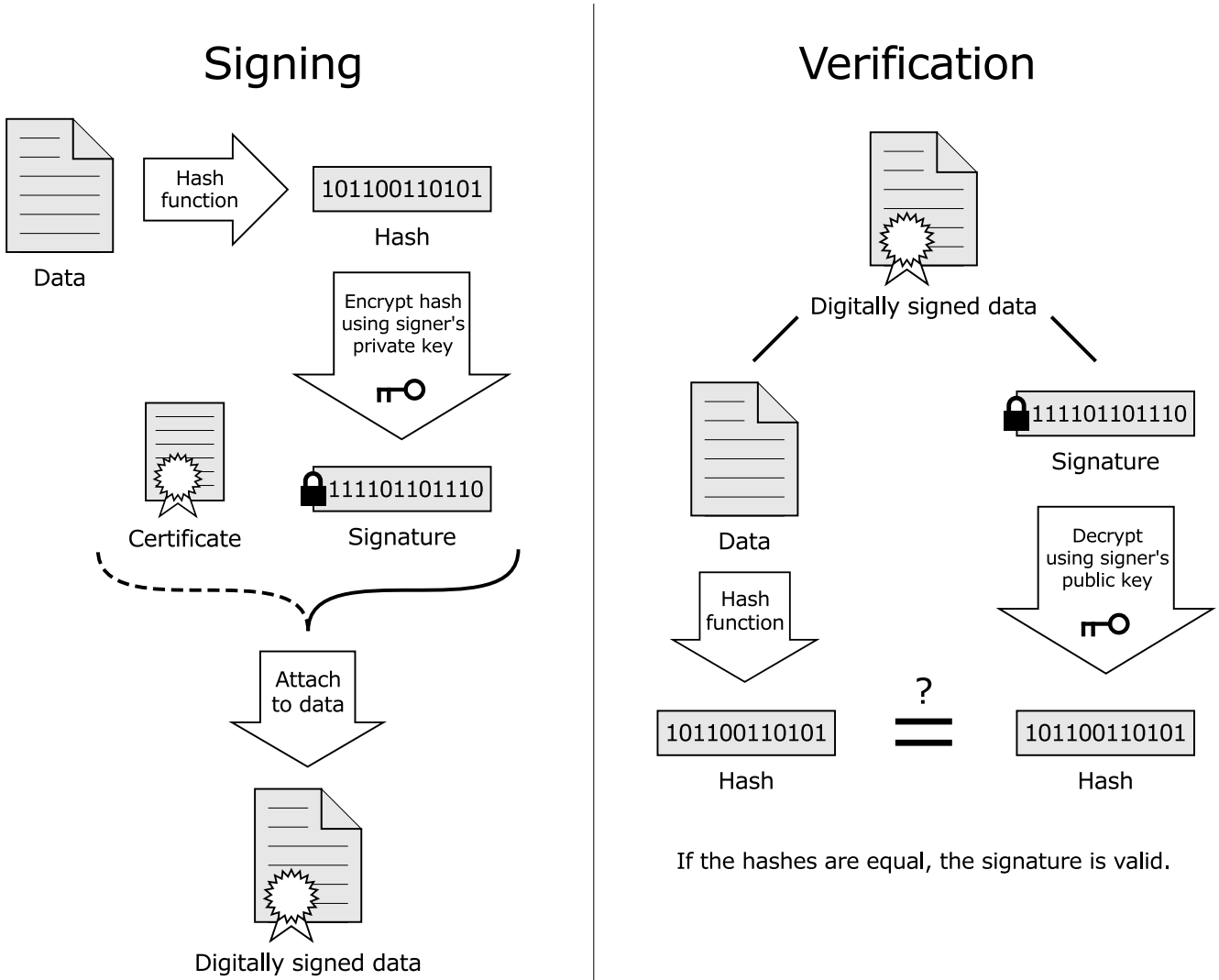
Felder in X.509 (vereinfacht):



Quelle: Microsoft: Digital Certificates. `https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/cc962029(v=technet.10)?redirectedfrom=MSDN` (Zugriff am 13.9.2022), 2012.

[2] Microsoft: Digital Certificates. `https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/cc962029(v=technet.10)?redirectedfrom=MSDN` (Zugriff am 13.9.2022), 2012.

Digitale Signaturen mit Zertifikaten



Quelle: Acdx: Diagram illustrating how a simple digital signature is applied and verified. `https://commons.wikimedia.org/wiki/File:Digital_Signature_diagram.svg` (Zugriff am 13.9.2022), 2012.

Zertifizierungsstellen

- Zertifizierungsstellen (engl. *certificate authorities*, CAs)
  - wird vertraut, Zertifikate auszustellen
  - überprüfen die Identität des Besitzers oder der Entität
  - haben ihr eigenes Schlüsselpaar (von jedem Akteur wird angenommen, dass er das CA-Schlüsselpaar kennt)
- Beschränkungen bei nur einer CA
  - Unwahrscheinlich, dass **jeder** der CA vertraut
  - Single point of failure: Was bei Kompromittierung tun?
  - Wie den CA-Schlüssel *sicher* herausfinden?
- Lösung: Mehrere CAs verwenden (mehrere Möglichkeiten; Beispiel: CA-Hierarchien)

Zertifizierungsstellenhierarchien I

- Zertifizierungsstellenhierarchien (CA-Hierarchien)
  - Ausstellen von Zertifikaten ist an Sub-CA(s) delegiert
  - Mehrstufige Zertifikate von der CA zu(r) Sub-CA(s) zur Entität

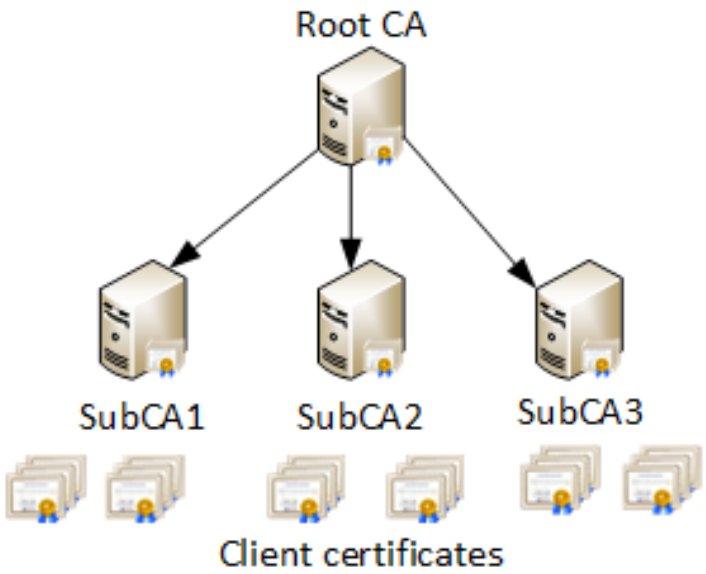
  $\rightarrow$ Hierarchisches Vertrauensmodell
- Zweistufiges Beispiel (mehr Stufen möglich):
  - CA stellt Zertifikat für Sub-CA (SCA) aus: $C_{SCA \rightarrow CA}$

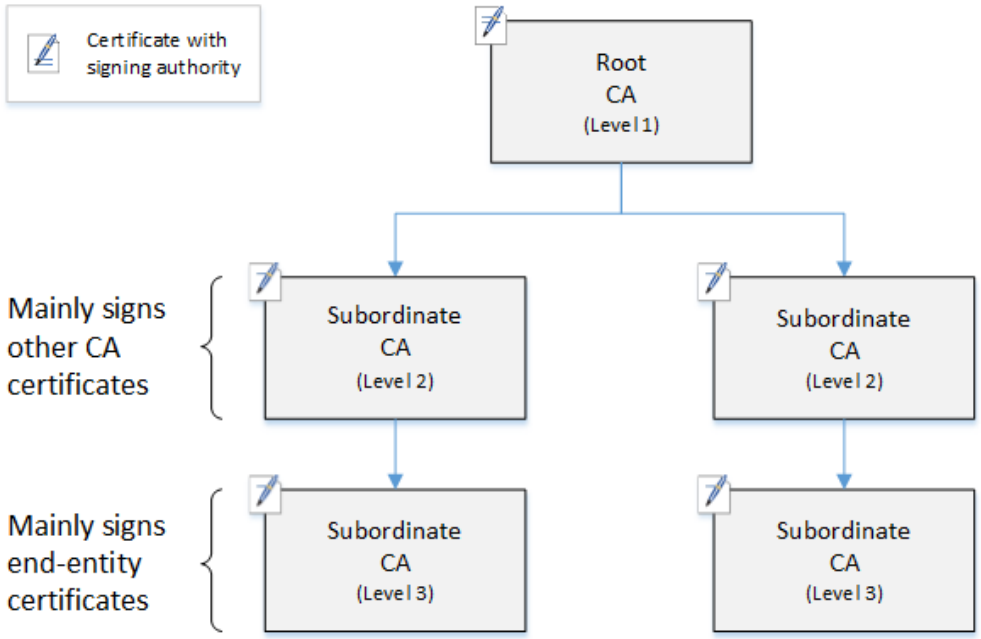
  $\rightarrow$ Sub-CA und ihr öffentlicher Schlüssel werden von CA signiert
  - Sub-CA stellt Zertifikat für Alice aus: $C_{A \rightarrow SCA}$

  $\rightarrow$ Bob kann Zertifikat von Alice durch Zertifikatskette $(C_{A \rightarrow SCA}, C_{SCA \rightarrow CA})$ überprüfen

  $\rightarrow$ Bob muss beiden – der Sub-CA und der CA – vertrauen!

Zertifizierungsstellenhierarchien II



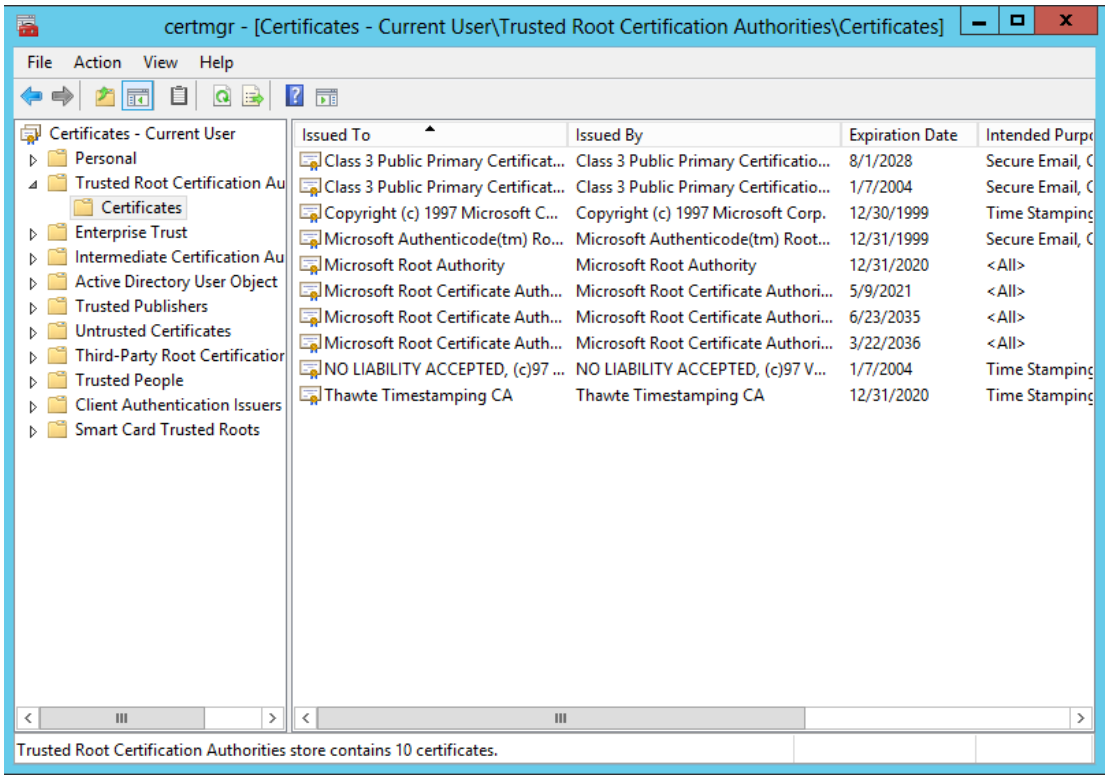


Quellen: Podāns, V.: Certificate Chaining Engine — how it works. `https://www.sysadmins.lv/blog-en/certificate-chaining-engine-how-this-works.aspx` (Zugriff am 13.9.2022), 2011; Amazon Web Services, Inc.: Designing a CA hierarchy. `https://docs.aws.amazon.com/acm-pca/latest/userguide/ca-hierarchy.html` (Zugriff am 13.9.2022), 2022.

Zertifizierungsstellenhierarchien III

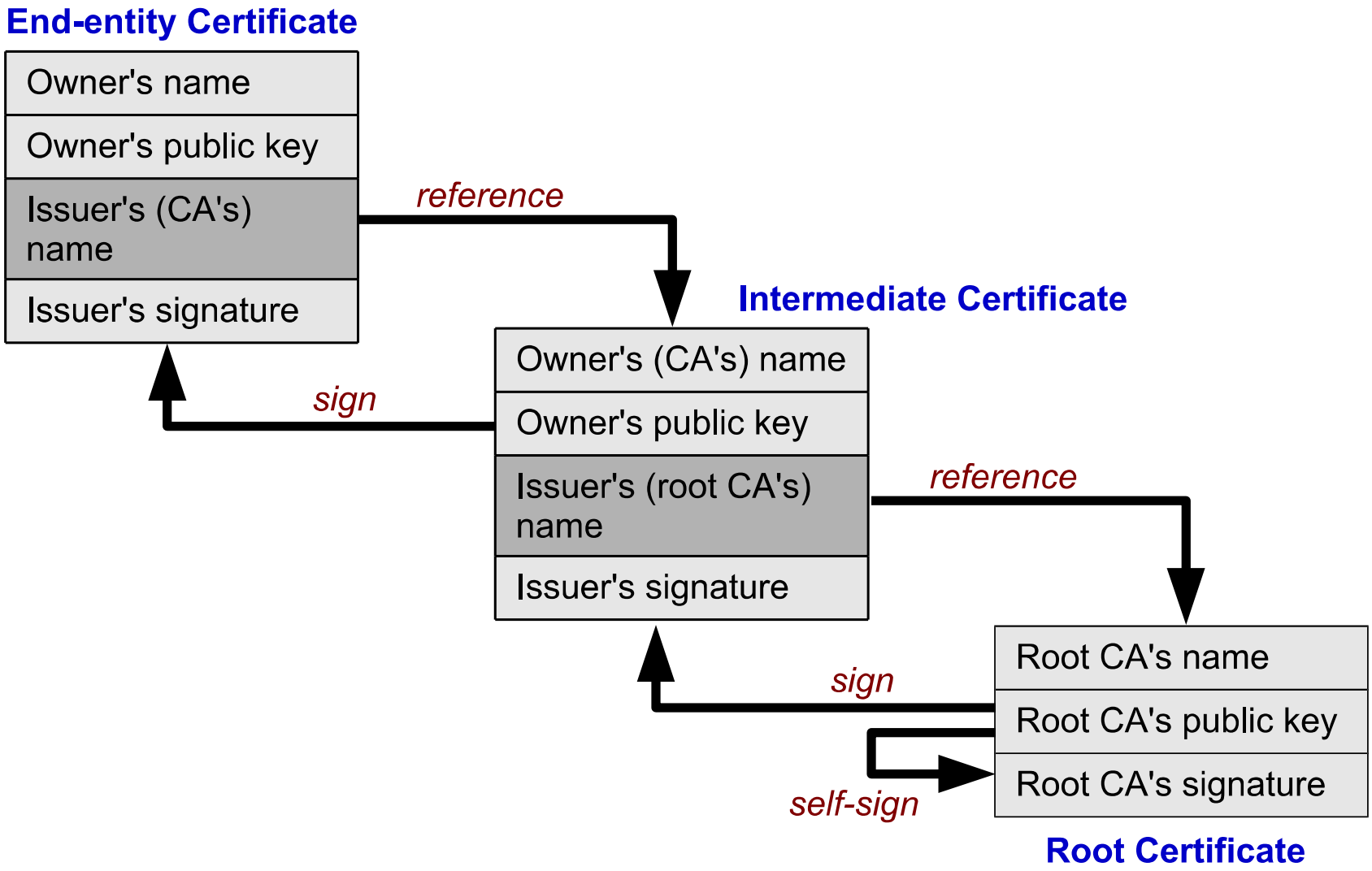
- Stammzertifizierungsstelle(n) (CA(s) auf höchster Hierarchieebene)
  - signieren ihr(e) eigenes/n Zertifikat(e) selbst
  - sind in Webbrowsern, Betriebssystemen etc. vorinstalliert

  → Hohes Maß an Vertrauen erforderlich



Quelle: pdubs: Where to get root CA certificates for Windows Server now that Microsoft no longer updates them? `https://serverfault.com/q/541922` (Zugriff am 13.9.2022), 2013.

Zertifizierungsstellenhierarchien IV



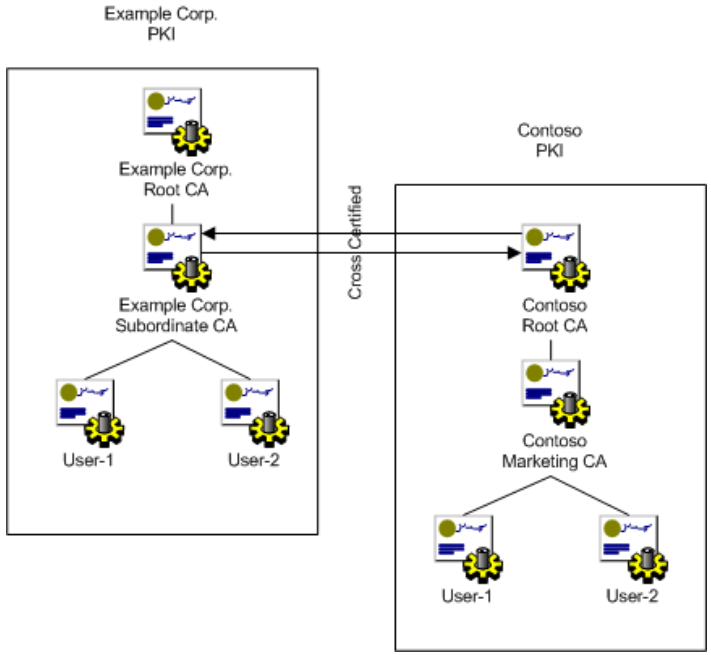
Quelle: Yanpas: Chain of trust SSL certificate. `https://commons.wikimedia.org/wiki/File:Chain_of_trust.svg` (Zugriff am 13.9.2022), 2017.

Praktische Belange I

- Es gibt keine universelle PKI
  - Verschiedene Anwendungsfälle/Organisationen haben ihre eigenen PKIs
  - → Benutzer/Entitäten haben verschiedene Schlüssel (und Zertifikate) für jeden Fall
  - → Benutzer/Entitäten verwenden mehrere PKIs zur selben Zeit
- Zurückziehen von Zertifikaten
  - Geheime Schlüssel werden gestohlen, kompromittiert, gehen verloren
  - → Möglichkeit, das entsprechende Zertifikat so schnell wie möglich zurückzuziehen
  - Linderung: Zertifikatsperrlisten (engl. *certificate revocation lists*, CRLs) und/oder Online-Zertifikatsüberprüfung durch CA oder andere vertrauenswürdige Partei
- Ablauf von Zertifikaten
  - Zertifikat hat ein(en) Gültigsdatum(sbereich)
  - → Zertifikat wird außerhalb dieses Datumsbereichs ungültig
  - → Zertifikat muss (regelmäßig) erneuert werden
  - Bei kurzen Gültigkeitszeiträumen ist keine CRL notwendig

Praktische Belange II

- Cross-Zertifizierung (CAs signieren gegenseitig ihre Zertifikate)
  - → Mehrere gültige Zertifikatsketten/-pfade existieren
  - → Robuster, wenn best. Zertifikate ablaufen oder zurückgezogen werden

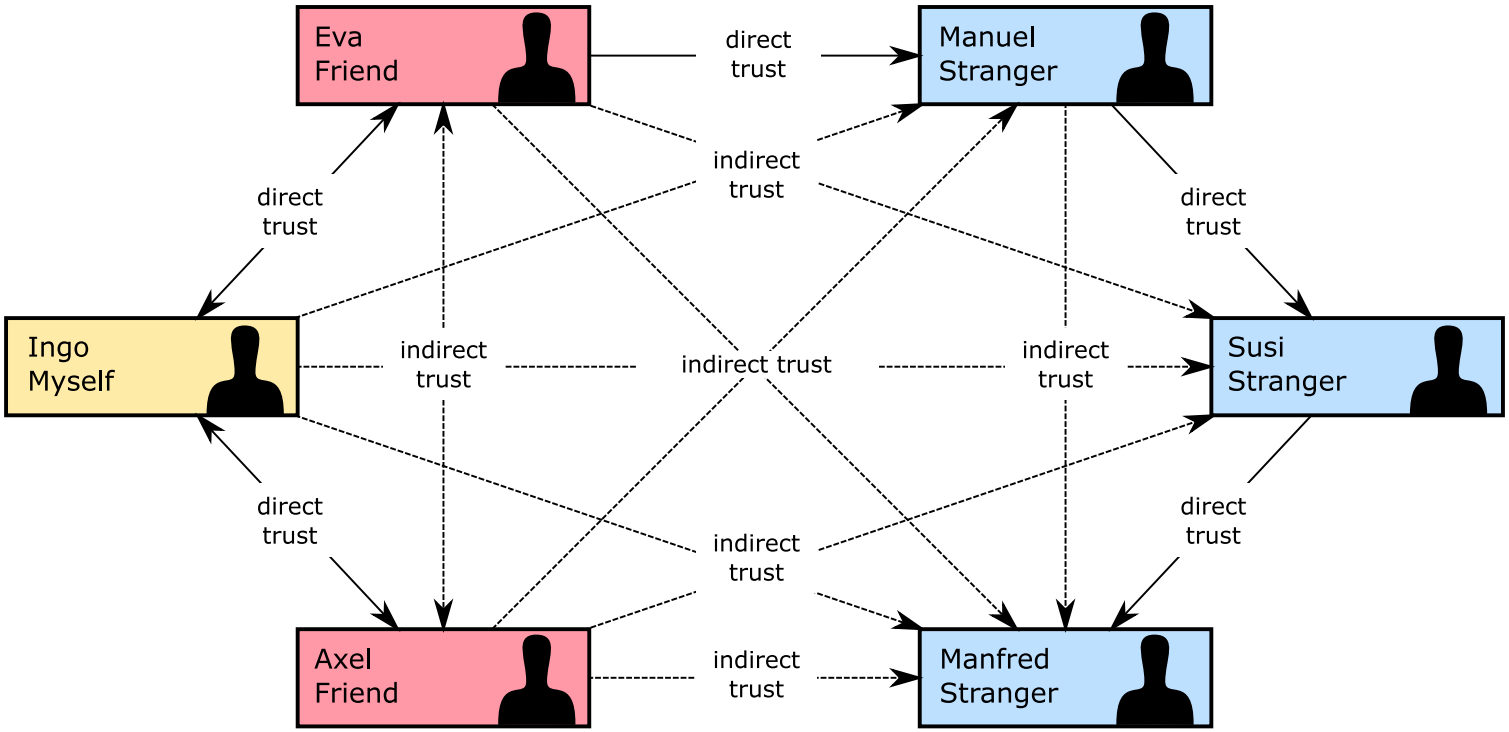


Quelle: Microsoft, Inc.: Cross Certification.
`https://docs.microsoft.com/en-us/windows/win32/seccertenroll/about-cross-certification` (Zugriff am 13.9.2022), 2021.

[3] SSLTrust: Understanding Certificate Cross-Signing.
`https://www.ssltrust.com.au/blog/understanding-certificate-cross-signing` (Zugriff am 13.9.2022), 2022.

Alternative Infrastrukturkonzepte

- Web of Trust
  - Jeder kann Zertifikate ausstellen (es gibt keine zentralen Stellen)
  - Jede Person oder Entität, die ein Zertifikat überprüft, muss entscheiden, ob sie einem beliebigen einzelnen Aussteller (direkt oder indirekt) vertraut



Quelle: Kku: Web of Trust network with levels of trust. `https://commons.wikimedia.org/wiki/File:Web_of_Trust-en.svg` (Zugriff am 13.9.2022), 2019.

Danke für die Aufmerksamkeit!

Fragen?