Kapitel 4: Asymmetrische Kryptographie

4: Asymmetrische Kryptographie

Public Key oder asymmetrische Kryptographie

Bei der asymmetrischer Kryptographie werden von Nutzern Schlüsselpaare (**pk**, **sk**) generiert. Dieses bestreht aus den folgenden Teilen.

- Public Key **pk** (öffentlicher Schlüssel)
- Secret Key **sk** (privater/geheimer Schlüssel)

- Der öffentliche Schlüssel wird weitergeben oder publiziert,
- der private Schlüssel ist ein Geheimnis und darf nicht preisgegeben werden.

Analogie

Briefkasten

Historie

ca. 1971: Ein Mitarbeiter des britischen Geheimdienstes erfindet die “non-secret” Kryptographie
(wurde bis Ende der 90-er Jahre geheimgehalten)

ca. 1974: "Merkle Puzzles"

ca. 1976: Diffie und Hellman

ca. 1977: Rivest, Shamir, Adleman (RSA)

seit ca. 1990: zunehmende kommerzielle Bedeutung der asymmetrischen Kryptographie

Hauptanwendungsgebiete

- Schlüsselaustausch
  - Zwei Teilnehmer generieren jeweils ein Schlüsselpaar.
  - Aus den beiden Schlüsselpaaren wird ein symmetrischer Schlüssel berechnet.
- Verschüsselung
  - Mit dem öffentlichen Schlüssel wird verschlüsselt
  - Mit dem privaten Schlüssel wird entschlüsselt
- Digitale Signaturen
  - Mit dem privaten Schlüssel wird signiert
  - Mit dem öffentlichen Schlüssel wird verifiziert

4.1: Modulare Arithmetik

- Uhr: Stunden mod 24, Minuten mod 60, Sekunden mod 60, ...
- Rechnerarithmetik: mod 2^w, $w \in \{8, 16, 32, 64\}$
- Prüfziffern mod 10 oder mod 11
- ...

Division mit Rest

Satz 5.1

Seien $a \in \mathbb{N}_0$ und $n \in \mathbb{N}$. Dann gibt es eindeutig bestimmte $q \in \mathbb{N}_0$, $r \in \{0, \ldots, n-1\}$ mit

$$a = n \cdot q + r.$$

Entsprechend definieren wir die Operationen "div" und "mod" durch

$$a \operatorname{div} n = q \quad \text{und} \quad a \bmod n = r$$

(das ganzzahlige Teilen und die Berechnung des Restes).

Division mit Rest

Beispiele

- $10 \operatorname{div} 7 = 1$
- $10 \bmod 7 = 3$
- $33 \operatorname{div} 4 = 8$
- $33 \bmod 4 = 1$
- $7 \operatorname{div} 5 = 1$
- $7 \bmod 5 = 2$

Restklassen

Modulo n sind alle Werte $a + (i \cdot n)$ äquivalent.

Definition 5.1

Seien $n \in \mathbb{N}$ und $a \in \mathbb{Z}$. Die Restklasse von a modulo n ist

$$\{i \cdot n + a | k \in \mathbb{Z}\}$$

Ein Element einer Restklasse bezeichnet man auch als Repräsentant der Restklasse. Die natürlichen Repräsentanten sind die Zahlen

$$\{\mathbf{0}, \ldots, \mathbf{n-1}\}.$$

Die Menge aller Restklassen modulo n, geschrieben $\mathbb{Z}_n$, bildet, zusammen mit der Addition und der Multiplikation, den Restklassenring mod n.

Restklassen

Beispiel

- $\mathbb{Z}_3$ besteht aus den folgenden Restklassen
  - Restklasse '0' : $\{\ldots, -12, -9, -6, -3, \mathbf{0}, 3, 6, 9, 12, \ldots\}$
  - Restklasse '1' : $\{\ldots, -11, -8, -5, -2, \mathbf{1}, 4, 7, 10, 13, \ldots\}$
  - Restklasse '2' : $\{\ldots, -10, -7, -4, -1, \mathbf{2}, 5, 8, 11, 14, \ldots\}$
- $\mathbb{Z}_5$ besteht aus den folgenden Restklassen
  - Restklasse '0' : $\{\ldots, -20, -15, -10, -5, \mathbf{0}, 5, 10, 15, 20, \ldots\}$
  - Restklasse '1' : $\{\ldots, -19, -14, -9, -4, \mathbf{1}, 6, 11, 16, 21, \ldots\}$
  - Restklasse '2' : $\{\ldots, -18, -13, -8, -3, \mathbf{2}, 7, 12, 17, 22, \ldots\}$
  - Restklasse '3' : $\{\ldots, -17, -12, -7, -2, \mathbf{3}, 8, 13, 18, 23, \ldots\}$
  - Restklasse '4' : $\{\ldots, -16, -11, -6, -1, \mathbf{4}, 9, 14, 19, 24, \ldots\}$

Bemerkungen

- Jede Restklasse hat genau einen natürlichen Repräsentanten $\mathbf{z \in \{0, \ldots, n-1\}}$, aber unendlich viele beliebige Repräsentanten $\mathbf{z} + i \cdot n \in \mathbb{Z}$.
- **Nie vergessen:** Mathematisch bedeutet eine Berechung in $\mathbb{Z}_n$, dass egal ist, wie eine Restklasse repräsentiert wird!
- $\mathbb{Z}_n$ besteht aus genau n Restklassen.
- In $\mathbb{Z}_n$ können wir rechnen wie in $\mathbb{Z}$ ($\rightarrow$ Rechenregeln mod n), bis auf die etwas andere $\rightarrow$ Kürzungsregel.
- Wenn klar ist, das wir für ein bestimmtes n in $\mathbb{Z}_n$ rechnen, können wir statt "$a \equiv b \pmod{n}$" auch "$a = b$" schreiben.

Die Kongruenz-Relation

Definition 5.2 (Kongruenz)

Gilt

$$a \bmod n = b \bmod n,$$

dann sind a und b kongruent modulo n und wir schreiben

$$a \equiv b \,(mod\, n).$$

Beispiel

- $10 \mod 7 = 17 \mod 7 \iff 10 \equiv 17 \pmod 7$

Eigenschaften mod n

Satz 5.2 (Äquivalenzrelationen)

Für alle $a, b, n \in \mathbb{N}$ gelten die folgenden Eigenschaften

Reflexivität: $a \equiv a \pmod{n}$

Symmetrie: $a \equiv b \pmod{n} \iff b \equiv a \pmod{n}$

Transitivität: $(a \equiv b \pmod{n}) \land (b \equiv c \pmod{n}) \Rightarrow a \equiv c \pmod{n}$

Beispiel

- $10 \equiv 10 \pmod{7}$
- $10 \equiv 17 \pmod{7} \iff 17 \equiv 10 \pmod{7}$
- $(10 \equiv 17 \pmod{7} \land 17 \equiv 3 \pmod{7}) \implies 17 \equiv 3 \pmod{7}$

Rechenregel mod n

Satz 5.3 (Regel vom Vielfachen)

Für alle $a, b \in \mathbb{N}$ mit $a > b$ gilt: $(a \equiv b \pmod{n}) \iff n|(a-b)$.

Beispiele

- $17 \equiv 10 \pmod{7} \iff 7|(17-10) = 7|7$
- $37 \equiv 13 \pmod{5} \iff 5|(37-13) = 5|25$

Mehr Rechenregeln mod n

Satz 5.4

Rechenregeln mod n

1. $(a + b) \bmod n = ((a \bmod n) + (b \bmod n)) \bmod n$
2. $(a - b) \bmod n = ((a \bmod n) - (b \bmod n)) \bmod n$
3. $(a \cdot b) \bmod n = ((a \bmod n) \cdot (b \bmod n)) \bmod n$
4. $a^d \bmod n = (a^{d-x} \cdot a^x) \bmod n = ((a^{d-x} \bmod n) \cdot (a^x \bmod n)) \bmod n$ *(für $x \leq d$)*

Mehr Rechenregeln mod *n*

Beispiel

- $(100 + 90) \bmod 11 = ((100 \bmod 11) + (90 \bmod 11)) \bmod 11 = (1 + 2) \bmod 11 = 3$
- $(100 - 90) \bmod 11 = ((100 \bmod 11) - (90 \bmod 11)) \bmod 11 = (1 - 2) \bmod 11 = 10$
- $(100 \cdot 90) \bmod 11 = ((100 \bmod 11) \cdot (90 \bmod 11)) \bmod 11 = (1 \cdot 2) \bmod 11 = 2$
- $3^5 \bmod 7 = ((3^2 \bmod 7) \cdot (3^3 \bmod 7)) \bmod 7 = (2 \cdot 6) \bmod 7 = 5$

Konsequenz

Folgerung

Bei Berechnungen modulo n kann man Zwischenergebnisse (alle Summen, alle Differenzen und alle Produkte) auf Werte in $\{0, \ldots, n-1\}$ reduzieren.

4.2: Diffie-Hellmann Schlüsselaustausch

Alice und Bob wollen einen gemeinsem Schlüssel austauschen und gehen dabei wie folgt vor.

- Alice generiert sich ein Schlüsselpaar
- Bob generiert sich ein Schlüsselpaar
- Alice sendet ihren öffentlichen Schlüssel an Bob
- Bob sendet seinen öffentlichen Schlüssel an Alice
- Bob berechnet aus seinem geheimen und Alice öffentlichen Schlüssel den gemeinsamen Schlüssel **K**
- Alice berechnet aus ihrem geheimen und Bobs öffentlichen Schlüssel den gemeinsamen Schlüssel **K**

Mathematische Beschreibung

Öffentliche Parameter (Setup)

- Primzahlen p und q mit $p = 2q + 1$ und $q > 2^{2047}$
- $g \in \mathbb{N}$ mit $1 < g < p$

Schlüsselaustausch

- Alice wählt einen zufälligen geheimen Schlüssel $0 < \mathbf{a} < p$
- Alice berechnet ihren öffentlichen Schlüssel $\mathbf{A} = g^{\mathbf{a}} \bmod p$
- Bob wählt einen zufälligen geheimen Schlüssel $0 < \mathbf{b} < p$
- Bob berechnet seinen öffentlichen Schlüssel $\mathbf{B} = g^{\mathbf{b}} \bmod p$
- Alice berechnet $\mathbf{K} = \mathbf{B}^{\mathbf{a}} \bmod p$
- Bob berechnet $\mathbf{K} = \mathbf{A}^{\mathbf{b}} \bmod p$
- Es gilt: $B^a \bmod p = g^{ba} \bmod p = g^{ab} \bmod p = A^b \bmod p$
- Optional wird $\mathbf{K}$ gehasht, d.h. $H(\mathbf{K})$, um einen AES-Schlüssel zu generieren.

Beispiel (in winzigen Zahlen)

0. $p = 11$, $g = 6$,
1. $\mathbf{a} = 4$, $\mathbf{A} = 6^{\mathbf{4}} \bmod 11 = 9$
2. $\mathbf{b} = 3$, $\mathbf{B} = 6^{\mathbf{3}} \bmod 11 = 7$
3. Die Berechnung des geheimen Schlüssels:
   - $\mathbf{B}^{\mathbf{a}} \equiv 7^{\mathbf{4}} \equiv \mathbf{3} \pmod{11}$,
   - $\mathbf{A}^{\mathbf{b}} \equiv 9^{\mathbf{3}} \equiv \mathbf{3} \pmod{11}$,
   - $g^{\mathbf{ab}} \equiv 6^{12} \equiv \mathbf{3} \pmod{11}$.

- Alice und Bob haben sich auf **3** als Geheimnis geeinigt.
- Ein Angreifer müsste $\boxed{g^{\mathbf{ab}} \bmod p}$ berechnen, obwohl er weder **a** noch **b** kennt.

Spezielle Potenz-Rechenregel

Satz 5.5 (Kleiner Satz von Fermat / Potenz Rechenregel)

Sei p Primzahl, $g \in \mathbb{Z}_p^*$ und $a \in \mathbb{N}$, dann gilt:

$$g^{p-1} \bmod p = 1.$$

Daraus folgt die spezielle Potenz-Rechenregel

$$g^a \bmod p = g^{a \bmod (p-1)} \bmod p.$$

Beispiel: $a = 12, g = 6, p = 11$

$$6^{12} \equiv 6^{12 \bmod 10} \equiv 6^2 \equiv 36 \equiv 3 \pmod{11}$$

Ist der DH-Schlüsselaustausch sicher?

Man betrachte die folgenden beiden Probleme:

- Diskreter Logarithmus (DL):
  Geg.: $\mathbf{X} = g^{\mathbf{x}} \pmod{n}$; Ges.: $\mathbf{x}$
- Diffie-Hellman-Problem (DH): Gegeben **A** und **B**, berechne $g^{\mathbf{ab}}$
- Man beachte: Wenn man effizient Diskrete Logarithmen berechnen kann, dann kann man erst recht das Diffie-Hellman Problem effizient lösen.
- Ist das Diffie-Hellman Problem dagegen nicht effizient lösbar, dann kann ein Angreifer, der nur **A** und **B** kennt, den geheimen Schlüssel $g^{\mathbf{ab}}$ nicht berechnen.
- Es wird *vermutet*: DL- und DH-Problem sind *nicht effizient lösbar*.
- Achtung: DH ist nicht sicher gegen aktive Angreifer. ($\rightarrow$ Tafel)

Libsodium

- Libdsodium ist eine moderne kryptographische Bibliothek.
- Einfache API im Gegensatz zu OpenSSL.
- Fork von `NaCl` (Prof. Dr. Dan Bernstein).
- Portabel: Linux, BSD, iOS, Android und Windows.
- Unterstützt asymmetrischen Schlüsselaustausch.
- `https://libsodium.gitbook.io/doc/key_exchange`
- Mehr Informationen: `https://libsodium.gitbook.io/`

Libsodium: Schlüsselaustausch

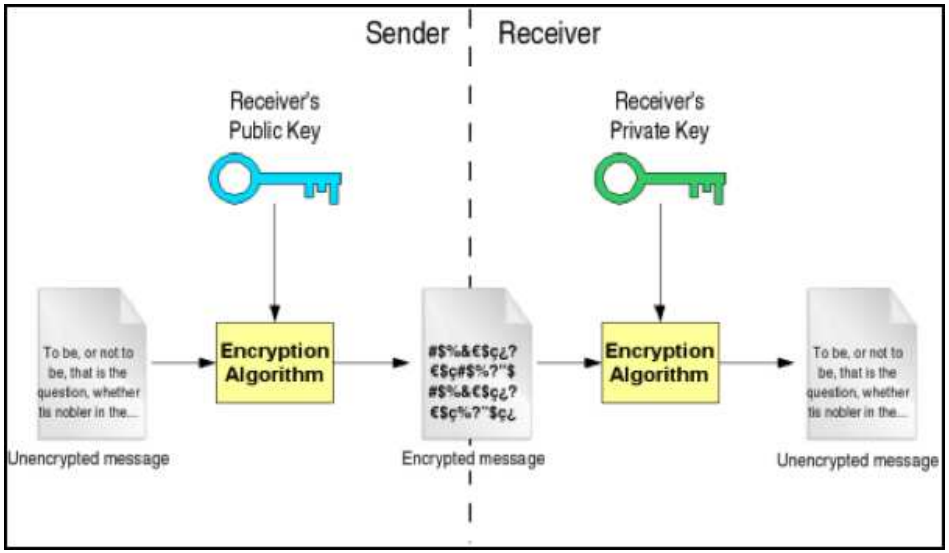
```
...
#define PUB_KEYLEN     crypto_kx_PUBLICKEYBYTES
#define SEC_KEYLEN     crypto_kx_SECRETKEYBYTES
#define SESSION_KEYLEN crypto_kx_SESSIONKEYBYTES

byte_t  *client_key_exchange(int socket) {
    byte_t local_pk[PUB_KEYLEN];
    byte_t local_sk[SEC_KEYLEN];
    byte_t rxkey[SESSION_KEYLEN];
    byte_t txkey[SESSION_KEYLEN];
    byte_t remote_pk[PUB_KEYLEN];
    byte_t *key = malloc(SESSION_KEYLEN);

    crypto_kx_keypair(local_pk, local_sk);
    send_pk2(local_pk, socket);
    receive_pk(remote_pk, socket);

    crypto_kx_client_session_keys(rxkey, txkey, local_pk,
                                  local_sk, remote_pk);
    for(int i=0; i< SESSION_KEYLEN; i++)
         key[i] = txkey[i]^rxkey[i];
     return key;
}
```

4.3: Asymmetrische Verschlüsselung



- Empfänger generiert ein Schlüsselpaar (**sk**, **pk**) und publiziert **pk**.
- Sender verschlüsselt seine Nachricht mit **pk**.
- Empfänger entschlüsselt die Nachricht mit **sk**.

RSA Kryptosystem

- Bekannteste Baustein der asymmetrischen Kryptographie
- Benannt nach seinen Entwicklern:
  - Ronald **R**ivest,
  - Adi **S**hamir
  - Leonard **A**dleman)
- Schlüssel von einem Paar großen Primzahlen p und q ab.
- Sicherheit beruht darauf, der Faktorisierug von großen Zahlen.
- Achtung: Der naive Einsatz von RSA ist unsicher.

RSA-Verschlüsselungsverfahren

Vorbereitung

1. Wähle zufällig *große* Primzahlen **p** und **q** (mind. 1024 Bit)
2. Berechne $\mathbf{n} = \mathbf{pq}$
3. Berechne $\varphi(\mathbf{n}) = (\mathbf{p} - 1)(\mathbf{q} - 1)$
4. Wähle $\mathbf{e} \in \mathbb{Z}^*_{\varphi(\mathbf{n})}$, d.h., ggT$(\mathbf{e}, \varphi(\mathbf{n})) = 1$
5. Berechne **d** mit

$$\mathbf{ed} \bmod \varphi(\mathbf{n}) = 1$$

Anmerkung

Ein solches **d** existiert, weil ggT$(\mathbf{e}, \varphi(\mathbf{n})) = 1$ gilt.

Eulersche Phi-Funktion φ

Definition 5.3 (Eulersche Phi-Funktion)

Sei $n \in \mathbb{N}$, dann ist $\varphi(n)$ die Anzahl an teilerfremden Zahlen $i \in \mathbb{N}$ mit $i < n$. D.h. es gilt:

$$\varphi(n) = \left|\{i = 1, \ldots, n-1 : \ ggT(i, n) = 1\}\right|.$$

Rechenregeln:

1. Sei p eine Primzahl dann gilt: $\varphi(p) = p - 1$
2. Sei $N = pq$ mit $p \neq q$ Primzahl, dann gilt: $\varphi(N) = (p-1)(q-1)$

Eulersche Phi-Funktion φ

Beispiele

- $\varphi(8) = |\{1, 3, 5, 7\}| = 4$
- $\varphi(31) = 31 - 1 = 30$ (Rechenregel 1)
- $\varphi(35) = (7 - 1)(5 - 1) = 24$ (Rechenregel 2)
- $\varphi(77) = (7 - 1)(11 - 1) = 60$ (Rechenregel 2)

Allgemeine Potenz-Rechenregel

Satz 5.6 (Euler)

Für $a, b, c \in \mathbb{N}$ und $ggT(a, c) = 1$ gilt:

$$a^b \bmod c = a^{b \bmod \varphi(c)} \bmod c.$$

Beispiel: $a = 7, b = 12, c = 15$

Es gilt $\varphi(15) = (3-1) \cdot (5-1) = 8$.

Damit gilt: $7^{12} \equiv 7^{12 \bmod 8} \equiv 7^4 \equiv 49 \cdot 49 \equiv 4 \cdot 4 \equiv 16 \equiv 1 \pmod{15}$.

Das RSA-Kryptosystem

- Klartextmenge = Chiffretextmenge = $\mathbb{Z}_{\mathbf{n}} = \{0, 1, \ldots n-1\}$
- Schlüssel: Tripel $(\mathbf{e}, \mathbf{d}, \mathbf{n})$; davon öffentlich: $(\mathbf{e}, \mathbf{n})$
- Verschlüsselung:

$$C \leftarrow M^{\mathbf{e}} \bmod \mathbf{n}$$

- Entschlüsselung:

$$M \leftarrow C^{\mathbf{d}} \bmod \mathbf{n}$$

Beobachtung

Es gilt:

$$C^{\mathbf{d}} \equiv (M^{\mathbf{e}})^{\mathbf{d}} \equiv M^{\mathbf{ed}} \bmod \varphi(\mathbf{n}) \equiv M^1 \equiv M \pmod{\mathbf{n}}$$

Beispiel

- Wir wählen $\mathbf{p} = \mathbf{13}$, $\mathbf{q} = \mathbf{11}$ und berechnen $\mathbf{n} = \mathbf{pq} = \mathbf{143}$.
- Es ist $\varphi(\mathbf{n}) = (\mathbf{p} - 1)(\mathbf{q} - 1) = \mathbf{120}$.
- Wir wählen $\mathbf{e} = \mathbf{7}$; insbesondere: ggT$(\mathbf{7}, \mathbf{120}) = 1$.
- Für $\mathbf{d} = \mathbf{103}$ gilt: $\mathbf{ed} = 721 = 6 \cdot 120 + 1 \equiv 1 \pmod{\mathbf{120}}$.
- Verschlüsseln des Klartextes $M = 5$:

$$C \equiv 5^{\mathbf{7}} \equiv 78125 \equiv 47 \pmod{\mathbf{143}}$$

- Entschlüsseln von $C = 47$:

$$M \equiv 47^{\mathbf{103}} \equiv 5 \pmod{\mathbf{143}}.$$

Sicherheit von RSA

Die Sicherheit des RSA-Systems beruht auf der (unbewiesenen) Vermutung, dass es schwierig ist, **n** zu faktorisieren (wenn **p** und **q** groß genug sind).

Anekdote: Frank Cole widerlegt 1903 eine fast 200 Jahre alte Vermutung von Mersenne:

> *Obwohl er die Sonntage dreier Jahre benötigte, um die Faktoren von* $\mathbf{2^{67} - 1}$ *zu finden, konnte er innerhalb weniger Minuten, ohne weitere Worte darüber zu verlieren, ein großes Publikum davon überzeugen, daß diese Zahl keine Primzahl war, indem er einfach die Arithmetik der Berechnungen aufschrieb:*
>
> $$\mathbf{2^{67} - 1 = 193707721 \cdot 761838257287}$$

Faktorisierung

Man könnte auf die Idee kommen, *eine Liste aller Primzahlen bis zu einer bestimmten Größe zu erstellen*. Um n zu faktorisieren, muss man nur probieren, n durch alle Primzahlen bis $\sqrt{n}$ zu teilen.

- Der Ansatz funktioniert grundsätzlich!
- Nur gibt es zu viele Primzahlen. Sei, z.B. N das Produkt zweier 1024-bit Primzahlen. Die Anzahl dieser Primzahlen übersteigt

$$2^{1023}/2^{11} = 2^{1012} \approx 4.4 \cdot 10^{303}.$$

- Selbst wenn wir 10^{100} Primzahlen pro Sekunde erzeugen könnten, bräuchten wir $4.4 \cdot 10^{203}$ Sekunden für das Erzeugen der Liste.
- Es gibt bessere Faktorisierungsalgorithmen, aber unser Universum ist nur etwa $4.3 * 10^{17}$ Sekunden jung

"Vorstellbare Angriffe" auf RSA

1. Faktorisieren von **n**. *(Gilt als extrem schwierig!)*
2. Berechnen von $\varphi(\mathbf{n})$. (Äquivalent zur Faktorisierung von **n**)
3. Berechnen der **e**-ten Wurzel modulo **n** ("RSA-Wurzel").
   (Vermutlich ebenso schwierig wie die Faktorisierung.)
4. Iteriertes Verschlüsseln des Kryptogramms.
   (Erfordert mit überwältigender Wahrscheinlichkeit astronomisch viele Iterationen

Naiver Einsatz von RSA

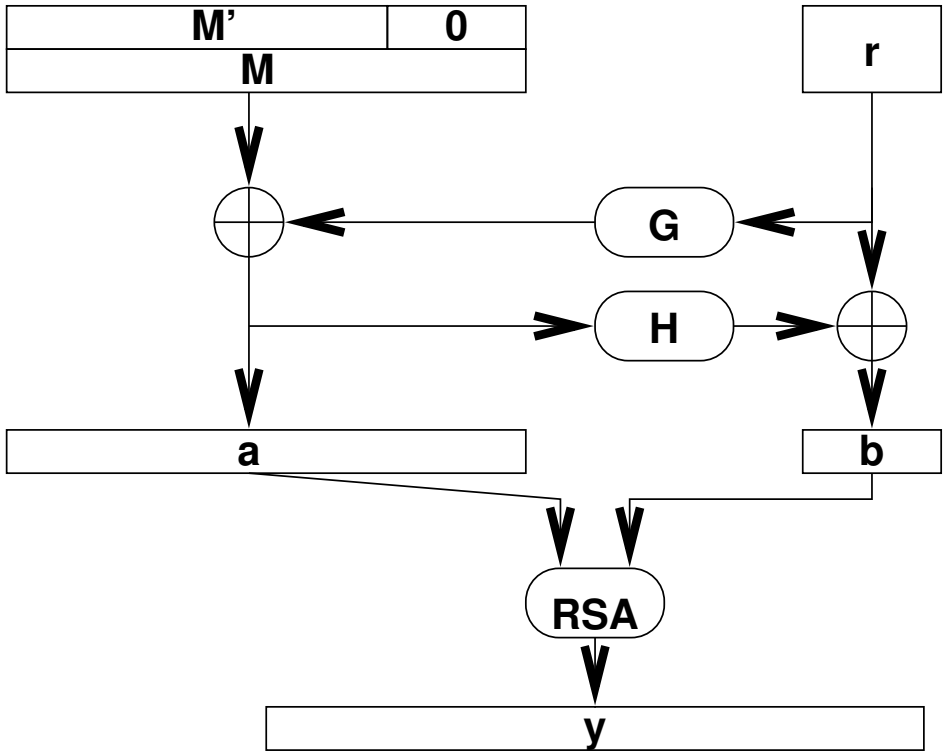
Der naive Einsatz von RSA ist sehr schwierig . . .

- RSA ist unsicher, bei Nachrichten mit niedriger Entropie. (**Warum?**)
- RSA ist unsicher, bei sehr kurzen Nachrichten. (**Warum?**)
- RSA nicht sicher gegen einseitige Angreifer (CPA). **Warum?**
- . . .

Optimal Asymmetric Encryption Padding (OAEP)

Sicheres RSA-Verschlüsselungsverfahren

Seien $G : \{0,1\}^* \to \{0,1\}^v$ und $H : \{0,1\}^* \to \{0,1\}^w$ zwei kryptographische Hashfunktionen. Dann kann eine Nachricht M mit $|M| \leq v$ unter RSA wie folgt, CPA-sicher, verschlüsselt werden.



Asymmetrische Verschlüsselungsverfahren

Vor- und Nachteile

+ Jeder Teilnehmer hat ein persönliches Geheimnis, das er mit niemandem teilen muss.
+ Kein Austausch von geheimen Schlüsseln notwendig
+ Spontane Kommunikation jederzeit möglich, ideal für offene Kommunikationssysteme.
+ Zahl der nötigen Schlüssel wächst linear mit der Zahl der Teilnehmer ($\rightarrow$ *Tafel*)

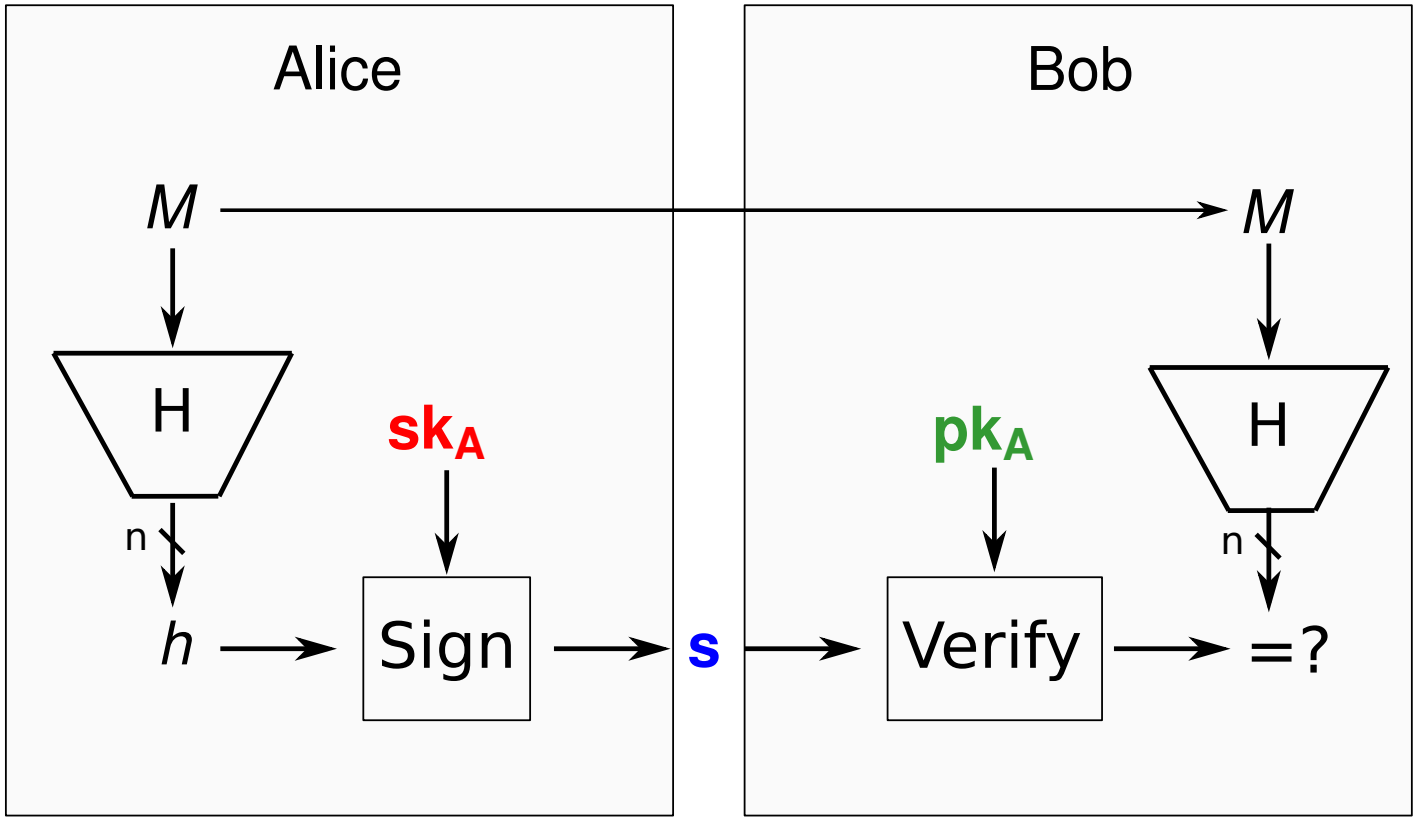
- Alle bekannten Verfahren sind sehr langsam (Faktor ca. 10.000 verglichen mit symmetrischen).
- Benötigte Schlüssellänge meist groß (Faktor 20 bis 40 verglichen mit symmetrischen).
- Schlüsselverwaltung bringt Komplikationen: öffentliche Schlüssel müssen authentisch sein

4.4: Digitale Unterschriften

Unterschrift "von Hand":

- Physikalische Verbindung mit dem unterschriebenen Dokument (beides steht auf dem gleichen Blatt).
- Fälschen erfordert einiges Geschick (wenn die Fälschung bei einer gründlichen Prüfung nicht auffallen soll . . .).
- Kopieren einer Unterschrift ist nicht möglich (Fotokopien und Faxe haben einen geringeren Beweiswert als Originale).

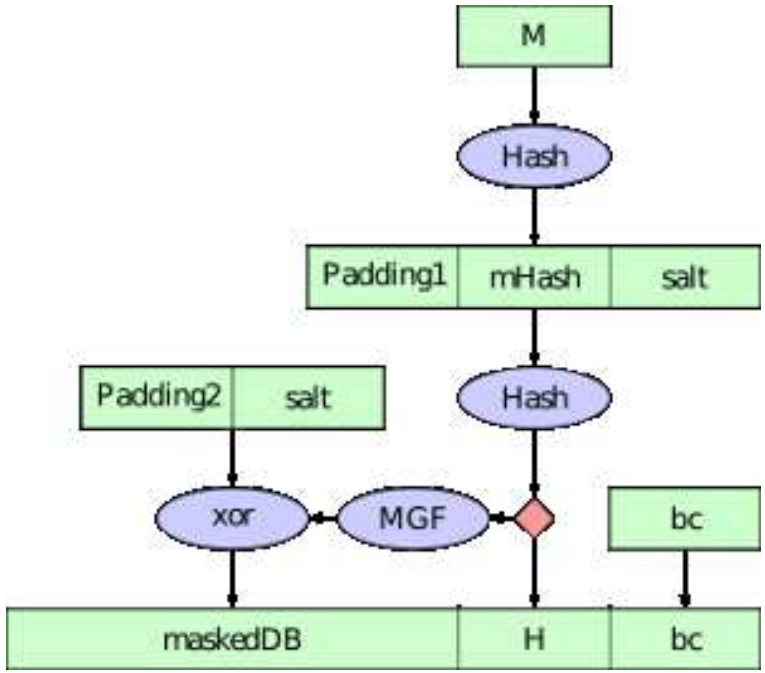
Funktionsweise: digitale Signatur



- Alice berechnet die digitale Signatur **s** aus dem Hashwert der Nachricht M mit Hilfe ihres privaten Schlüssels $\mathbf{sk_A}$.
- Bob verifiziert mit Hilfe Alice öffentlichem Schlüssel $\mathbf{pk_A}$ die Signatur **s**

Probabilistic Signature Scheme (PSS)

Sicheres RSA-Signierverfahren



Quelle: Johannes Böck `https://rsapss.hboeck.de/`

Alternative:

Full Domain Hash mit Hilfe von $cSHAKE256(M, |n|, , \text{"signature"})$ (NIST Special Publication 800-185).

Vertrauensfrage

Problem

Woher weiß Bob, dass der öffentliche Schlüssel $\mathbf{pk_A}$ wirklich zu Alice, und nicht zu Eve gehört.

- Der persönliche Austausch von Schlüsselmaterial ist nicht praktikabel.
- Bei einem elektronischen Schlüsselaustausch besteht oftmals die Gefahr eines Man-in-the-Middle-Angriffs.

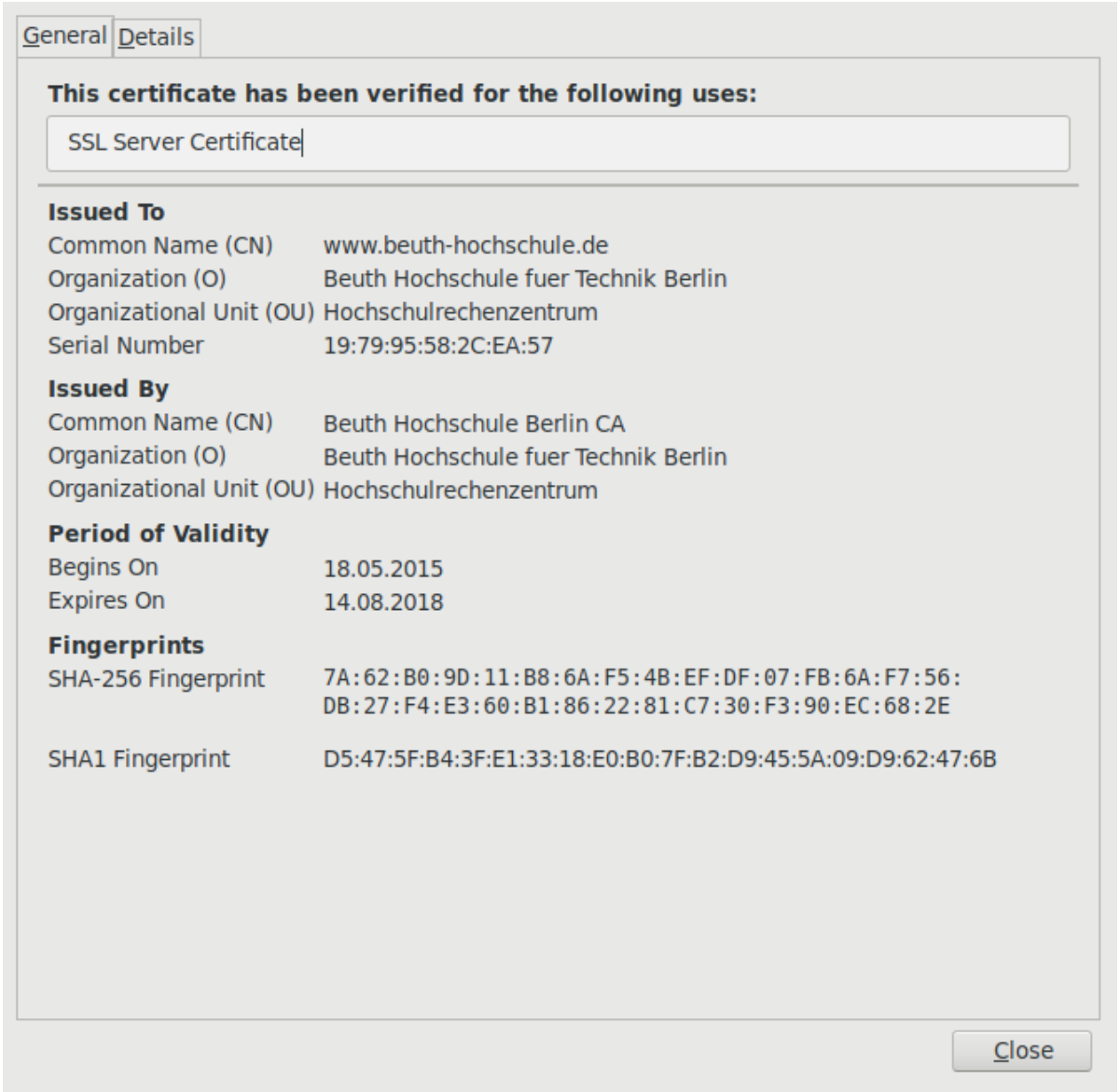
Lösung: Zertifikate

Ein Zertifikate ist ein Ausweis für einen öffentlichen Schlüssel.

Zertifikate

- Bei digitale Signaturen kommen häufig Zertifikate zum Einsatz.
- Zertifikat: Public Key + Metainformationen
  - ID
  - Name des Inhabers
  - Gültigkeitsdauer
  - Aussteller
  - ...
- Ein Zertifikat ist vom Aussteller digital signiert
- Das Signieren von Zertifikaten für Hostnamen ist ein Geschäftsmodell.
- Das Internet verwendet X.509 Zertifikate

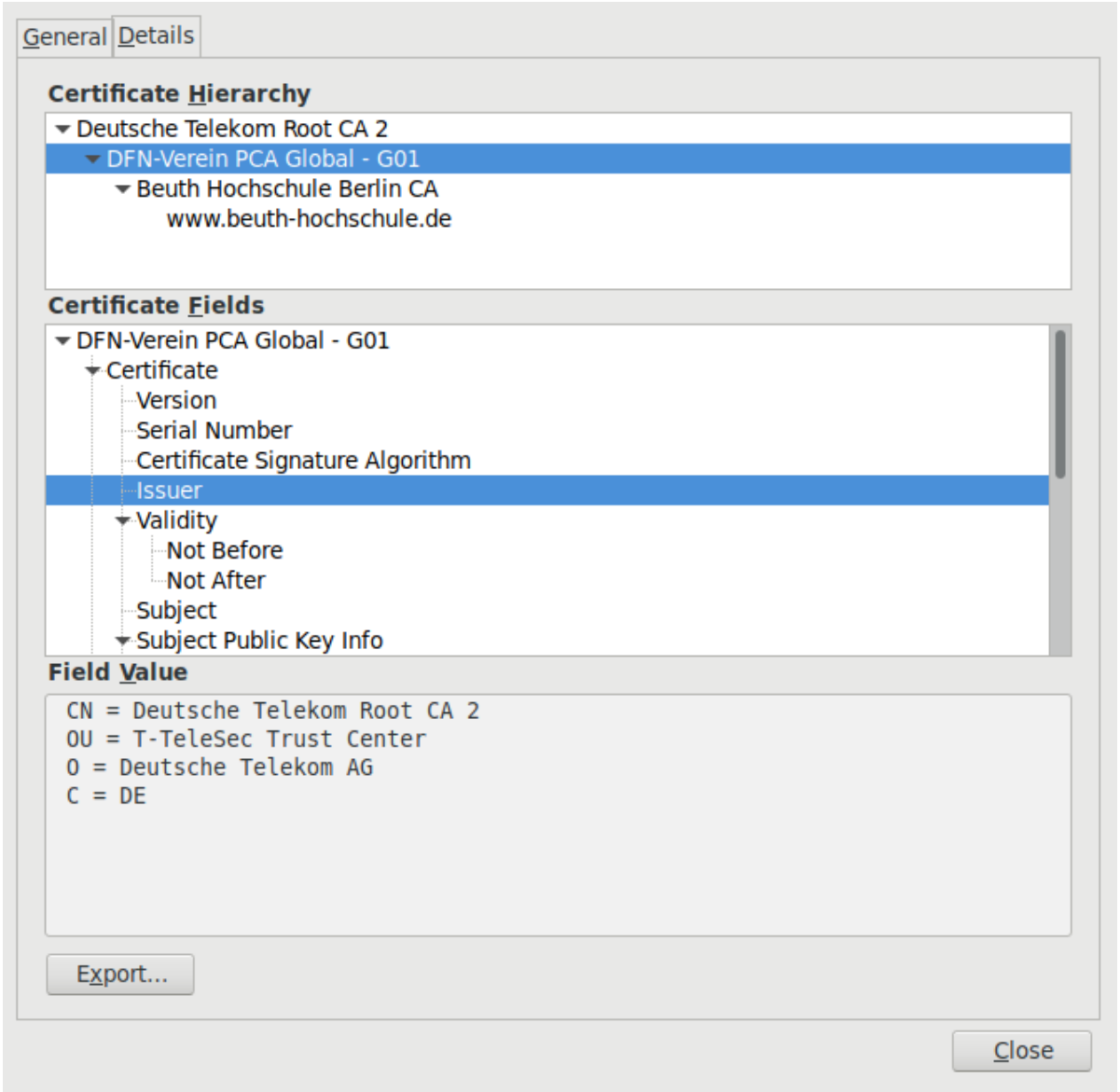
Beispiel für X.509 Zertifikat



Zertifikatsketten

- Ein Zertifikat ist das Ende einer Vertrausenskette.
- Ein Zertifikat ist meist von einer Certification Authorities (CA) ausgestellt.
- Die CA hat wiederum ein Zertifikat, welches von einer anderen. CA ausgestellt wurden.
- Die erste CA in der Kette wird Root-CA genannt. Das Zertifikat der Root-CA hat die Root-CA sich selbst ausgestellt. (*engl. self signed*).
- Damit Zertifikatsketten funktionieren muss der Root-CA vertraut werden.
- Programme wie Webbrowser haben eine Liste von Root-CAs welche sie blind vertrauen.

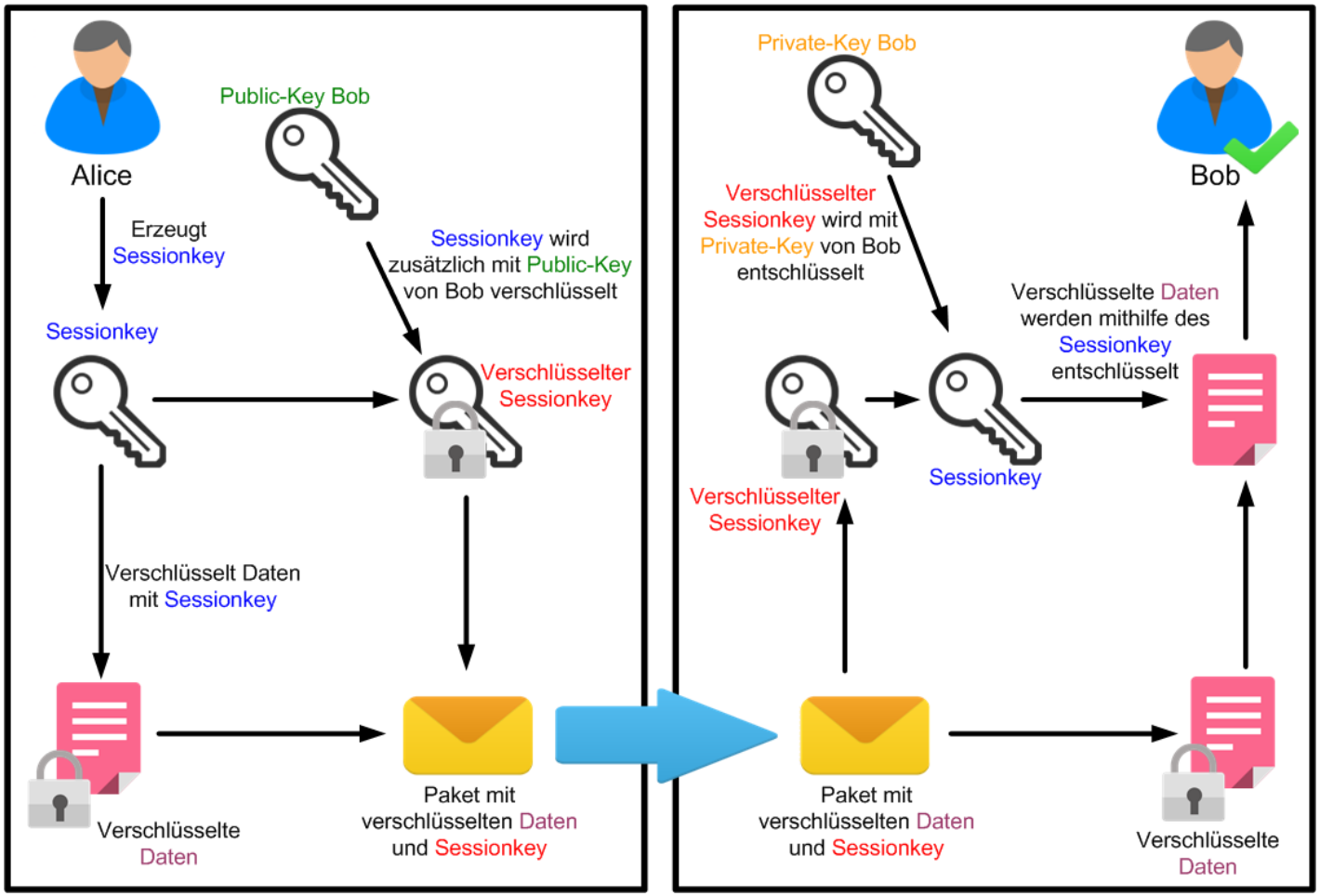
Beispiel: Zertifikatskette



4.5: Hybride Verschlüsselung

- Die Verschlüsselung von (großen) Nachrichten (Daten) mit einem asymmetrischen Verschlüsselungsverfahren PK-Scheme ist ineffizient.
- Besser Hybride Verschlüsselung
  - Nachricht mit einem symmetrischen AE-Scheme zu verschlüsseln
  - Symmetrischer Schlüssel mit PK-Scheme verschlüsseln
  - Zum Entschlüsseln der Daten muss zunächst der symmetrische Schlüssel entschlüsselt werden.
- Die meisten kryptographische Protokolle Nutzen hybride Verschlüsselung
- Pro Sitzung wird ein neuer symmetrischer Schlüssel ausgehandelt (Sitzungsschlüssel)

Visualisierung



Quelle: Dekocrypt (CC BY-SA 3.0)

Praxisrelevanz

Hybride Verschlüsselung ist allgegenwärtig.

- Verschlüsselung von Emails (S/MIME, PGP)
- Sichere Netzwerkverbindungen (IPSec)
- Webseiten (HTTPS, TLS)
- Remote Login (SSH)
- ...

Sicherheitsparameter

Schutz	**Symmetrisch**	**Asymmetrisch**
Schwach (wenige Wochen)	80	1024
Mittelfristig (bis 2028)	128	3072
Langfristig (bis 2068)	256	15360

(Quelle: `http://www.keylength.com/en/3/`)

4.6: Schlussbemerkungen

Sie sollten in der Lage sein, . . .

- . . . beispielhaft zu erläutern, wie public-key Kryptographie grundsätzlich funktioniert.
- . . . den DH-Schlüsselaustausch erläutern können.
- . . . die Funktionsweise von asymmetrische Verschlüsselungsverfahren kennen.
- . . . wissen wie digitale Signaturen funktionieren.
- . . . verstanden haben was Zertifikate sind.
- . . . verstanden haben welches Probleme Zertifikate lösen
- . . . verstanden haben wie hybride Verschlüsselung funktioniert