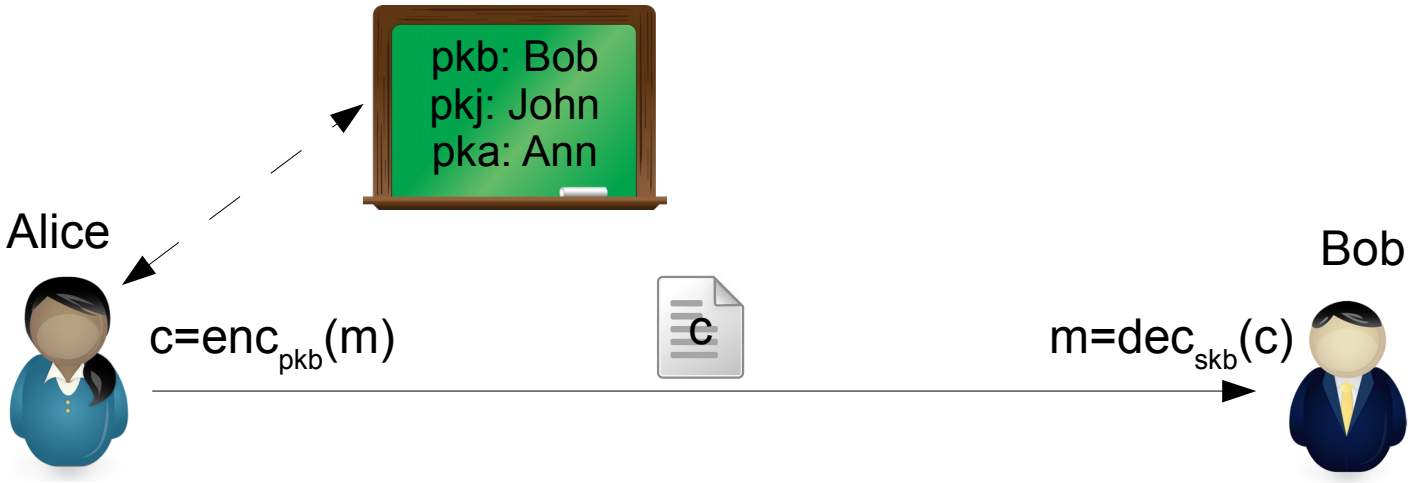
Themen zur Computersicherheit

Asymmetrische Chiffren

PD Dr. Reinhard Bündgen
buendgen@de.ibm.com

Asymmetrische Verschlüsselungsverfahren



- enc: eK x $\sum_1^*$ → $\sum_2^*$, dec: dK x $\sum_2^*$ → $\sum_1^*$
- Schlüssel (pk,sk) ∈ eK x dK mit pk ≠ sk
  - pk heißt öffentlicher Schlüssel (public key)
  - sk heißt privater Schlüssel (private / secret key)
  - sk kann aus pk nicht effizient berechnet werden
- Wenn Alice eine vertrauliche Nachricht an Bob senden will,
  - verschlüsselt sie die Nachricht mit Bobs öffentlichen Schlüssel & sendet sie an Bob
  - Bob entschlüsselt sie mit seinem privaten Schlüssel
- Verfahren
  - Verschlüsselung: Rivest-Shamir-Adleman (RSA)
  - Schlüsselaustausch: Diffie-Hellman (DH)

Primzahlen, teilerfremde Zahlen

- Sei p eine Primzahl
- **Z**/p**Z** = GF(p) ist ein endlicher (Galois-)Körper
  - **Z**/p**Z** ist eine additive Gruppe
  - {1, …, p-1} ist eine multiplikative Gruppe in **Z**/p**Z**
  - $x^{p-1} = 1 \bmod p$ für alle $0 < x < p$
- Primzahltests
  - kleine Primzahlen: Sieb des Eratosthenes
  - große Primzahlen: z.B. Rabin-Miller (probabilistisch)
- ggT, Teilerfremdheit
  - a, b sind teilerfremd, wenn ggT(a,b) = 1
  - erweiterter Euklidischer Algorithmus: (g,u,v) = EE(a,b) mit g=ggT(a,b) = u•a + v•b
  - Inverse mod p: EE(a,p) = $(1,a^{-1} \bmod p,v)$
  - ebenso gilt für beliebige a, n mit ggT(a,n) = 1: EE(a,n) = $(1, a^{-1} \bmod n,v)$
- Satz von Euler und Folgerungen
  - $\phi(n)$ ist die Zahl der zu n teilerfremden Zahlen kleiner n (Eulersche Phi-Funktion)
    - ggT(a,n) = 1 => $a^{\phi(n)} = 1 \bmod n$
    - => kleiner Fermat: p prim: $a^{p-1} = 1 \bmod p$ für alle $a < p$
    - (∗): p,q prim: $a^{k\phi(p\bullet q)+1} = a \bmod (p\bullet q)$ für alle a
- Wieviele Primzahlen gibt es?
  - unendlich viele
  - Primzahlen sind häufig: Es gibt ungefähr n/(ln n) Primzahlen kleiner n
  - Beispiel: es gibt ca 2^{1013} Primzahlen der Länge 1024 Bit

RSA

- seien p, q Primzahlen
- n = p • q
- ϕ (n) = (p-1) • (q-1) die Zahlen in **Z**/n**Z**, die zu p und q teilerfremd sind
- (n,e) ist ein öffentlicher Schlüssel, wenn e teilerfremd zu ϕ (n)
- (p,q,d) mit d = e^{-1} mod ϕ(n) ist der dazugehörige private Schlüssel
  - => e • d = 1 + k • ϕ(n)
- für Nachricht m < n gilt:
  - $\text{rsa-enc}_{(n,e)}(m) = m^e \bmod n$
  - $\text{rse-dec}_{(p,q,d)}(c) = c^d \bmod n$
  - $\text{rsa-dec}_{(p,q,d)}(\text{rsa-enc}_{(n,e)}(m)) = \text{rsa-dec}_{(p,q,d)}(m^e \bmod n) = (m^e)^d \bmod n = m^{ed} \bmod n = m \bmod n$
    - wegen Folgerung (∗) aus Satz von Euler:
      - (∗): p,q prim: $a^{k\phi(p \cdot q)+1} = a \bmod (p \cdot q)$ für alle a

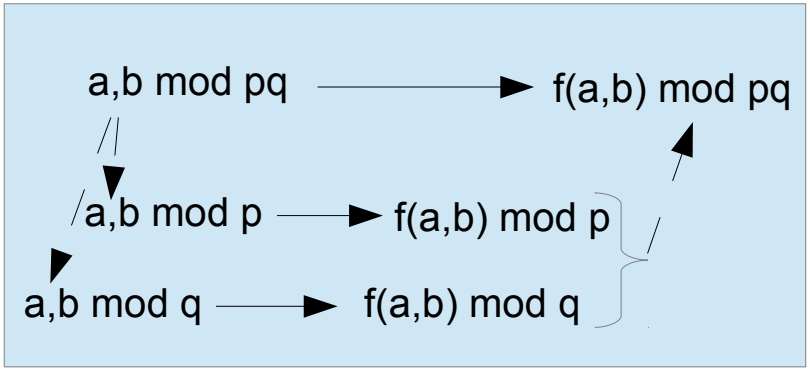
Beispiel

- p = 11
- q = 5
- n = p • q = 55
- ϕ (n) = (p-1) • (q-1) = 10 • 4 = 40
- e = 3
  - ggT(3,40) = 1
- 3 • d = 1 + k • 40
  - k = 1: 41 ist kein Vielfaches von 3
  - k = 2: 81 / 3 = 27 => d = 27
- öffentlicher Schlüssel (55,3)
- privater Schlüssel (11,5,27)

- Nachricht m = 50
- $enc_{(55,3)}(m) = 50^3 \bmod 55 = 40 \bmod 55$
- $dec_{(11,5,27)}(40) = 40^{27} \bmod 55 = 50 \bmod 55$

Der Chinesische Restesatz (Chinese Remainder Theorem / CRT)

- erste Erwähnung durch chin. Mathematiker Sun Tzu aus dem 1. Jahrhundert.
- CRT (Version mit 2 Primfaktoren)
  - Sei n = p • q mit p, q prim, dann gibt es zu zwei Zahlen a und b genau ein x $\in$ **Z**/n**Z** mit x = a mod p und x = b mod q
- Garners Formel
  - $x = (((a - b) \cdot (q^{-1} \bmod p)) \bmod p) \cdot q + b$
- Relevanz
  - eine Rechnung mod n kann ersetzt werden durch
    - zwei Rechnungen mod p bzw mod q und
    - einer Instanz von Garners Formel
  - lohnt sich wenn Rechnungsaufwand superlinear in der Zahlenlänge (ld n) ist
    z.B. Langzahlmultiplikation
    - naiv: $O((\text{ld } n)^2)$
    - Karazuba $O((\text{ld } n)^{\text{ld } 3})$
    - Schönhage-Strassen $O((\text{ld } n) \cdot (\log(\text{ld } n)) \cdot (\log (\log (\text{ld } n))))$
    - Fürer (2007); Harvey et al (2014): $O((\text{ld } n) \cdot (\log(\text{ld } n) \cdot 2^{3\log^*(\text{ld } n)}))$



- RSA: der private Schlüssel wir oft im CRT Format angegeben:
  - (p, q, d_p, d_q, q^{-1}) mit $d_p = e^{-1} \bmod (p-1) = d \bmod (p-1)$ und $d_q = e^{-1} \bmod (q-1) = d \bmod (q-1)$

Beispiel

- $d_p = 7 = 3^{-1} \bmod 10$, da $3 \cdot 7 = 21 = 1 \bmod 10$
- $d_q = 3 = 3^{-1} \bmod 4$, da $3 \cdot 3 = 9 = 1 \bmod 4$
- $q^{-1} = 9 \bmod 11$, da $5 \cdot 9 = 54 = 10 \bmod 11$

- $40^7 \bmod 11 = 7^7 \bmod 11 = 7 \cdot 49^3 = \bmod 11 = 7 \cdot 5^3 \bmod 11 = 35 \cdot 25 \bmod 11 = 6 \bmod 11$
- $40^3 \bmod 5 = 0 \bmod 5$

Garners Formel

- $x = (((a - b) \cdot (q^{-1} \bmod p)) \bmod p) \cdot q + b$

Also ergibt sich für 40^{27}

- für a = 6, b = 0:
- $((6 - 0) \cdot 9) \bmod 11) \cdot 5 + 0 = (54 \bmod 11) \cdot 5 = 10 \cdot 5 = 50$

Einschränkungen und Fallstricke

- Die Sicherheit von RSA beruht auf der Schwierigkeit große Zahlen zu faktorisieren
- Der numerische Wert INT(m) der Nachricht m muss kleiner als der Modulus n sein.
- Für kleines INT(m) und kleines e kann $INT(m)^e$ kleiner als n sein. Dann kann INT(m) durch Wurzelziehen in den natürlichen Zahlen berechnet werden.
- Verschiedene RSA Moduli dürfen keine gemeinsamen Teiler haben.
  - ggTs zu berechnen ist viel effizienter möglich als Zahlen zu faktorisieren
  - Nadia Henninger, Zakir Durumeric, Eric Wustrow, J. Alex Halderman: „Mining Your Ps and Qs: Detection of Widespread Weak Keys in Network Devices“ Proc. 21st USENIX Security Symposium, 2012
- Zufallsprimzahlgeneratoren müssen viele verschiedene Primzahlen generieren

Homomorphie Eigenschaft von RSA

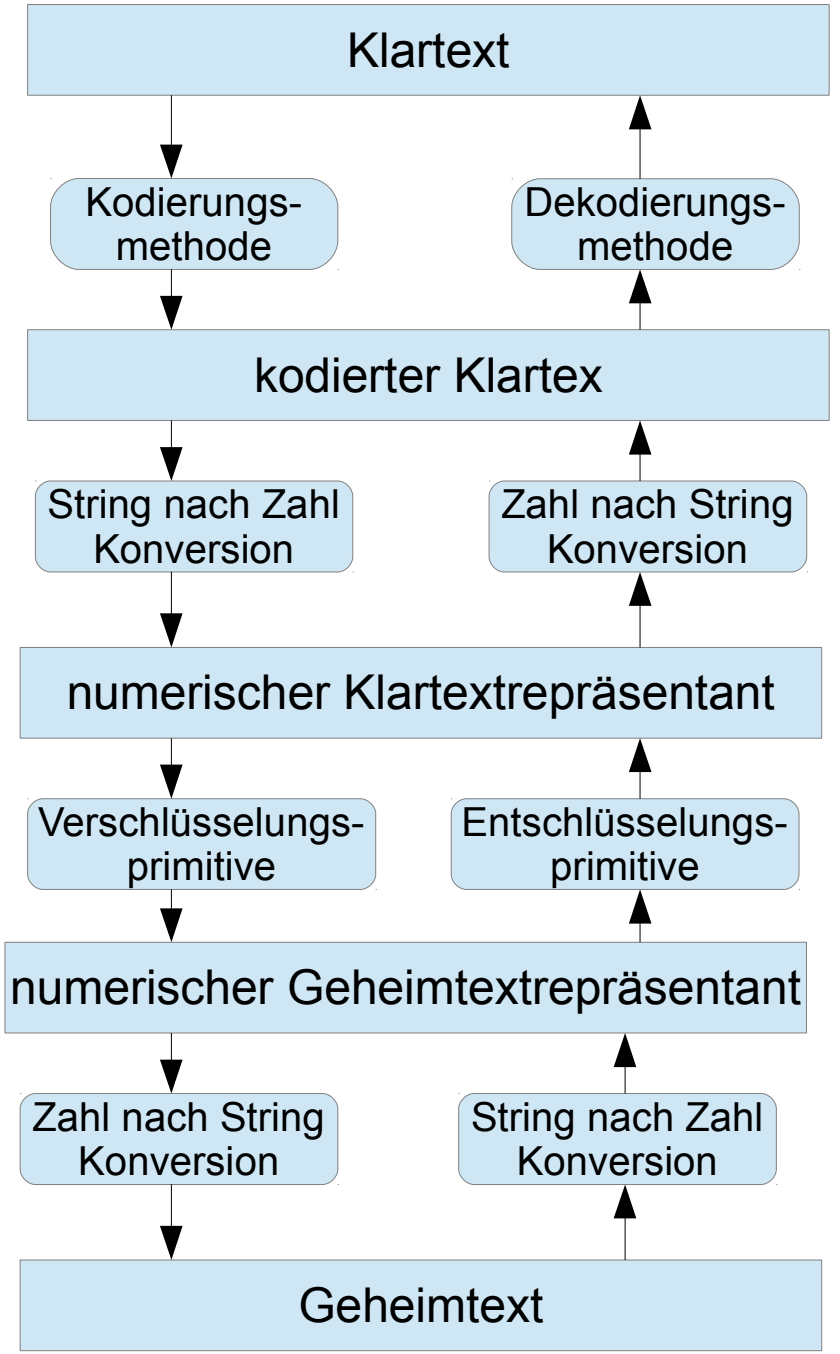
Es gilt

- $\text{rsa-enc}_{(n,e)}(m_1 \cdot m_2) = \text{rsa-enc}_{(n,e)}(m_1) \cdot \text{rsa-enc}_{(n,e)}(m_2) \bmod n$,
- denn
  - $\text{rsa-enc}_{(n,e)}(m_1 \cdot m_2)$
  - $= (m_1 \cdot m_2)^e \bmod n$
  - $= m_1^e \cdot m_2^e \bmod n$
  - $= (m_1^e \bmod n) \cdot (m_2^e \bmod n) \bmod n$
  - $= \text{rsa-enc}_{(n,e)}(m_1) \cdot \text{rsa-enc}_{(n,e)}(m_2) \bmod n$
- die RSA Verschlüsselung ist homomorph bzgl der Multiplikation modulo n
- die RSA Verschlüsselung ist nicht homomorph bezüglich der Addition oder xor

Verschlüsselung nach PKCS #1 v2.2

- beschreibt Verschlüsselungsschemata
  - RSAES-OAEP (Optimal Asymmetric Encryption Padding)
  - RSAES-PKCS-v1_5

  Sicherheit
  - RSAES-OAEP ist sicher gegen adaptive CCA Attacken, falls die Verschlüsselungsprimitive schwer (partiell) zu invertieren ist
  - RSAES-PKCS-v1_5 nicht für neue Anwendungen empfohlen, da CCA-Angriffe bekannt (Bleichenbacher-Attacke)



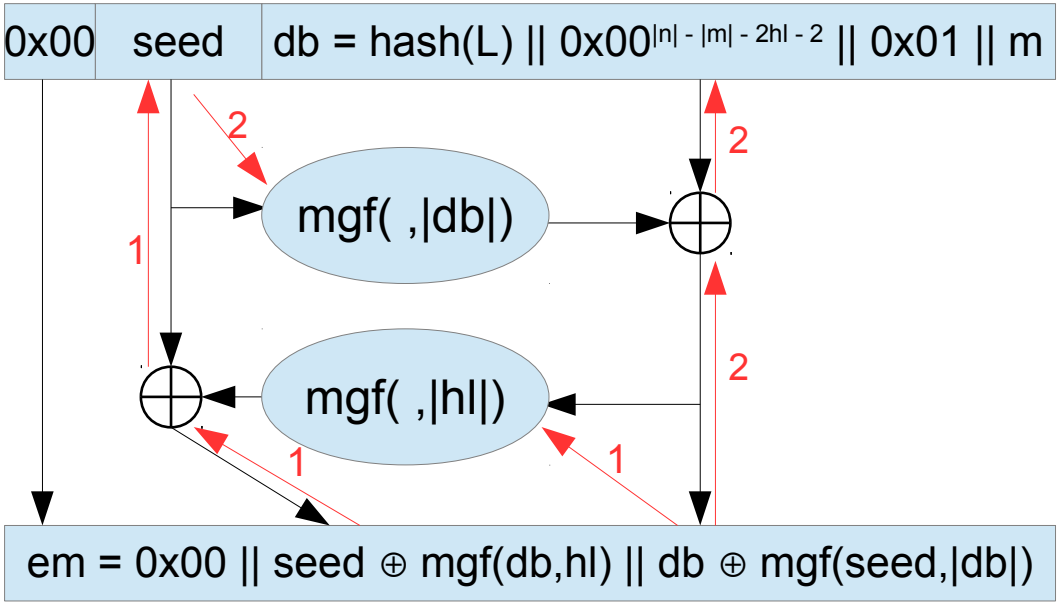
PKCS #1 Kodierungsoperationen

PKCS #1- v1_5 Kodierung

- Eingabe
  - Länge des Modulus: |n| in Bytes
  - Klartext: m mit |m| ≤ |n| - 11
- Ausgabe:
  - kodierter Klartext:
  - 0x00 || 0x02 || PS || 0x00 || m
  - mit PS besteht aus |n| - |m| - 3 pseudozufällig generierten Bytes ungleich 0x00
- Umkehrung
  - m beginnt nach zweiten 0x00 Byte

PKCS #1- v2.2 OAEP Kodierung

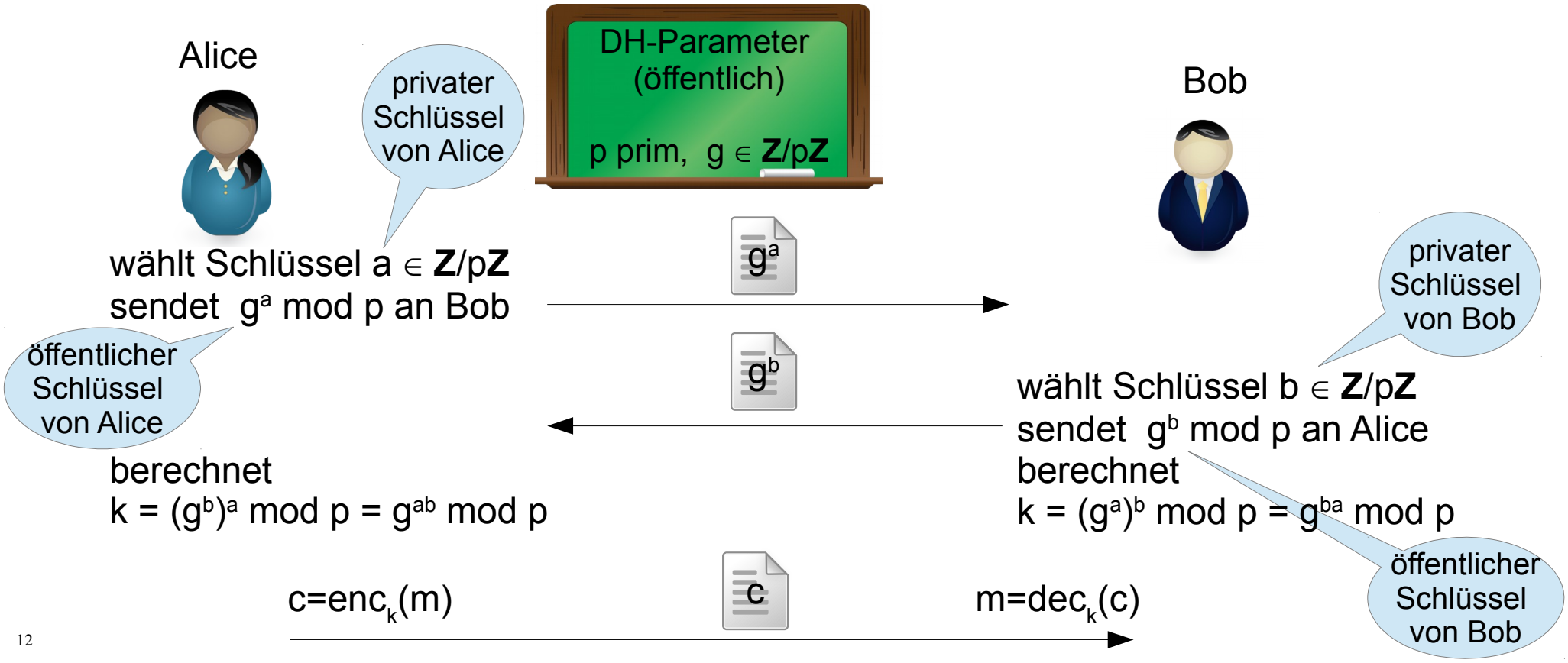
- Eingabe
  - Länge des Modulus: |n| in Bytes
  - Hashfunktion: hash: $\sum \rightarrow \sum^{hl}$
  - maskenerzeugende Funktion: mgf: $\sum^* \times \mathbf{N} \rightarrow \sum^*$
  - optionales Label: $L \in \sum^*$
  - seed: Zufallsbytestring der Länge hl



- Ausgabe
  - kodierter Klartext: em
- Umkehrung
  - seed = seed⊕mgf(db,hl) ⊕ mfg(db⊕mgf(seed,|db|),hl)
  - db = db⊕mgf(seed,|db|) ⊕ mgf(seed,|db|)
  - m beginnt nach dem ersten 0x01 Byte hinter Byte 2 • hl

Diffie-Hellman Schlüssel Austausch

- PKCS #3
- Sei p prim, dann heißt k der *diskrete Logarithmus* von n zur Basis a modulo p wenn a^k = n modulo p
- Es gibt kein effizientes Verfahren um für große Primzahlen p diskrete Logarithmen zu berechnen
- Protokoll: seinen p prim und g ∈ **Z**/p**Z** öffentlich



Schlüsselerzeugung

- Das gemeinsame Geheimnis $k = g^{ab} \bmod p$ muss nicht unbedingt Element des Schlüsselraums sein.
- Deshalb wird der eigentliche Schlüssel oft mit Hilfe einer (Schlüsselableitungsfunktion basierend auf einer) kryptographischen Hashfunktion von k abgeleitet.

Ein paar Begriffe aus der Gruppentheorie

- p prim, dann ist $Z_p^* = \{1, \ldots, p-1\}$ die multiplikative Gruppe modulo p
- für $a \in Z_p^*$ ist $\{a^1, \ldots, a^{p-1}\}$ die von a erzeugte Untergruppe von Z_p^*,
- $|\{a^1, \ldots, a^{p-1}\}|$ heißt die *Ordnung* von a,
- die Ordnung ist immer ein Teiler von p-1
- wenn a die Ordnung p-1 hat heißt a *primitives Element* von Z_p^*

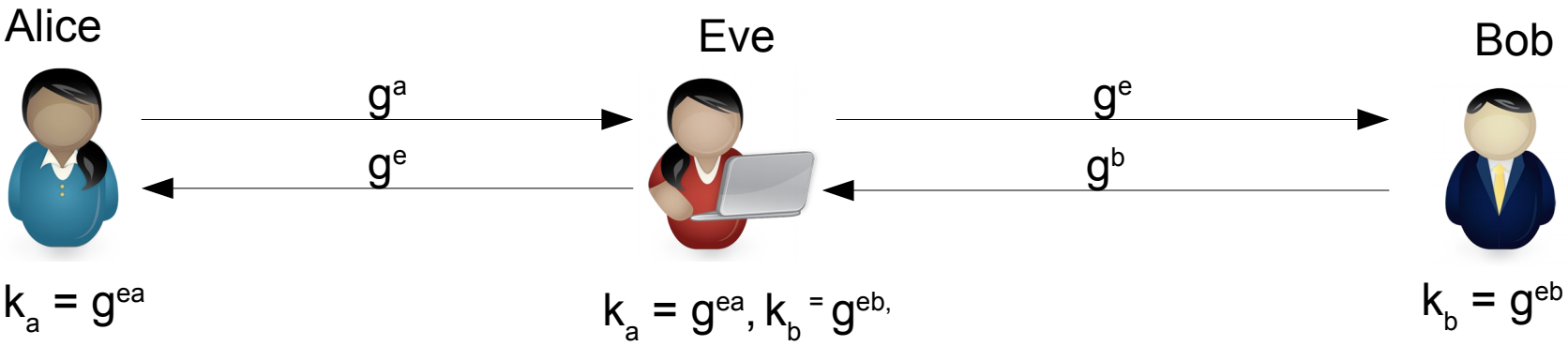
Die Wahl des DH-Parameters g

- { g^1 mod p, g^2 mod p, g^3 mod p, … } muss sehr große Menge sein
- wenn g primitives Element, dann ist |{g^i mod p | i < p }| = p-1
- Wie kann man verhindern, dass g mit kleiner Ordnung gewählt wird?
  - wähle p so, dass p-1 außer der 2 keine kleinen Teiler hat
  - sichere Primzahlen: p = 2 • q + 1 für q prim
    - dann hat Z_p^* nur Untergruppen der Ordnung 1, 2, q und 2 • q
    - Hälfte der Zahlen in {1, ...p-1} sind Quadratzahlen modulo p
    - alle Elemente der UG der Ordnung q sind Quadratzahlen modulo p
    - es gibt einen einfachen Test um festzustellen, ob h mod p eine Quadratzahl ist
    - nutze nur Elemente der UG mit Ordnung q:
    - => für $\alpha \in \{2, \ldots, p-2\}$ mit $\alpha^2 \neq 1 \bmod p$ und $\alpha^2 \neq p-1 \bmod p$, setze $g = \alpha^2$

- Effizienter Schlüsselaustausch
  - nutze g, das kleinere Untergruppe der Ordnung q erzeugt
  - dann gilt: $g^x \bmod p = g^{x \bmod q} \bmod p$
  - RFC2631, ANSI X9.42
    - wähle p = N • q + 1, mit q prim
      - wobei die Größe von q die Größe des erzeugbaren Schlüsselraum bestimmt
      - wähle g so, dass $g = \alpha^N \bmod p$, $g \neq 1 \bmod p$, $g \neq p-1 \bmod p$
    - die öffentlichen DH Parameter sind dann (p, q, g)

Angriffe auf DH

- Man in the middle Angriff:
  - Eve fängt Protokoll zwischen Alice und Bob ab.



  - Gegenmaßnahme: Signieren der Nachrichten (→ später in der Vorlesung)

- Manipulation der DH Parameter
  - z.B. g mit geringer Ordnung
  - Gegenmaßnahme:
    - verifizieren von p und g (bzw q)
      - Längentest von p und q
      - q teilt p?
      - $g \neq 1 \bmod p$, $g^q = 1 \bmod p$
    - verifizieren von empfangenem g^a, bzw g^b
      - z.B. für alle g^b muss gelten $1 < g^b \bmod p < p$ und $(g^b)^q = 1 \bmod p$

Statische und Ephemere DH Verfahren

- Wie lange sollen erzeugte Schlüssel benutzt werden?
  - für immer
  - nur für eine Nachricht
- Was passiert, wenn ein privater Schlüssel entdeckt wird?
  - Können dann alle alten Nachrichten entschlüsselt werden?
  - Können dann alle zukünftigen Nachrichten entschlüsselt werden?
- statisches DH Verfahren: Schlüssel bleibt konstant
  - static-static: Alice und Bob halten Schlüssel konstant
- ephemeres (flüchtig) DH Verfahren: Schlüssel ändern für jede Nachricht
  - static-ephemeral: Bob ändert den Schlüssel für jede Nachricht
  - ephemeral-static: Alice ändert den Schlüssel für jede Nachricht
  - ephemeral-ephemeral: Alice & Bob ändern die Schlüssel für jede Nachricht
  - → perfect forward privacy (PFP)

- Verschlüsselung:
  - wenn beim static-ephemeral DH Verfahren Alice ihren öffentlichen Schlüssel in einem öffentlichen Verzeichnis veröffentlicht, dann kann DH ähnlich wie RSA zum Verschlüsseln genutzt werden
  - sonst muss Alice ihren öffentlichen Schlüssel zusammen mit der verschlüsselten Nachricht an Bob schicken.

Varianten von DH: ECDH

- Elliptic Curve Diffie-Hellman (ECDH)
- Elliptische Kurve über endlichen Körper GF(p^n): $E = \{ (x,y) \in GF(p^n)^2 \mid y^2 = x^3 + ax + b \} \cup O$
- auf Elliptischen Kurven lässt sich eine (multiplikative) Gruppe beschreiben wobei
  - $(x_1,y_1) \bullet (x_2, y_2) = (x_3, y_3)$ mit
  - $x_3 = ((y_2-y_1)/(x_2-x_1))^2 - x_1 - x_2$; $y_3 = y_1 + ((y_2-y_1)/(x_2-x_1))) \bullet (x_1 - x_3)$ für $x_1 \neq x_2$ oder $y_1 \neq y_2$
  - $x_3 = ((3x_1^2 + a) / (2y_1))^2 - 2x_1$; $y_3 = -y1+((3x_1^2 + a) / 2y_1) \bullet (x_1-x_3)$ für $x_1 = y_1$ und $x_2 = y_2$
  - Potenzierung (bzgl. A in Gruppe): $A^n = A \bullet A \bullet \ldots \bullet A$ (n-faches Produkt)
  - (diskreter) Logarithmus zur Basis A von X: finde n so das $X = A^n$
- das Problem des diskreten Logarithmus bei elliptischen Kurven ist schwerer als bei Standard-DH → kleiner Modulus
- Domain Parameter (p, a, b, G, n, h)
  - $G \in E$ ist Generator, n Ordnung von G in E, h Kofaktor von G
- Schlüssel
  - privater Schlüssel: $d \in \{1, ..., n-1\}$ — Skalar
  - öffentlicher Schlüssel: $P = G^d \in E$ — Punkt
- Bemerkung: In der Literatur wird „•“ oft als Addition „+“ bezeichnet und dann die n-fache Addition durch ein skalares Produkt bezeichnet

Sind DH bzw EC Kryptographie sicher?

Gerücht: Die NSA kann DH knacken.

- Beobachtung. sehr viele Server nutzen die gleichen DH Parameter
  - Die Berechnung von $\log(g^a \bmod p)$ hängt zu einem großen Teil nur von p und g ab.
  - Möglicher Angriff: Vorberechnen von Teilrechnungen aller diskreten Logarithmen bezüglich häufig genutzter Parameter
    - Vermutung NSA hat Ressourcen um so bis zu 1024 bit Schlüssel zu knacken
  - das wäre zusammen mit einem Angriff, der beim Schlüsselaustauschprotokoll eine kleinen DH Modulus (z.B. Logjam) erzwingt ein effektiver Angriff auf DH
  - Literatur:
    - Adirian et al: „Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice“, https://weakdh.org/imperfect-forward-secrecy-ccs15.pdf

- „gezinkte“ Primzahlen &special number field sieve (SNFS) erlauben effiziente Logarithmenberechnung
  - Fried et al: „A kilobit hidden SNFS discrete logarithm computation“ http://eprint.iacr.org/2016/961
    - diskreter Log von 1024-Bit Primzahl in 2 Monaten (auf 3000 Kernen) geknackt
  - „gezinkte“ Primzahlen sind sehr selten, lassen sich aber konstruieren
  - Vorschlag (FIPS 186-4): veröffentliche mit p den Seed des PRNG der p erzeugt hat

Es gibt wohl einen (möglichen) Angriff auf DH und ECDH per Quantum Computing

- 2015: NSA empfiehlt
  - keine neuen Investitionen in ECC Verfahren zu tätigen
  - und statt dessen in Verfahren zu investieren, die gegen Quantum-Computing sicher sind
- diese NSA Empfehlung ist bei Wissenschaftlern umstritten
- Literatur:
  - Koblitz und Menezes: „A riddle wrapped in an enigma“, https://eprint.iacr.org/2015/1018.pdf