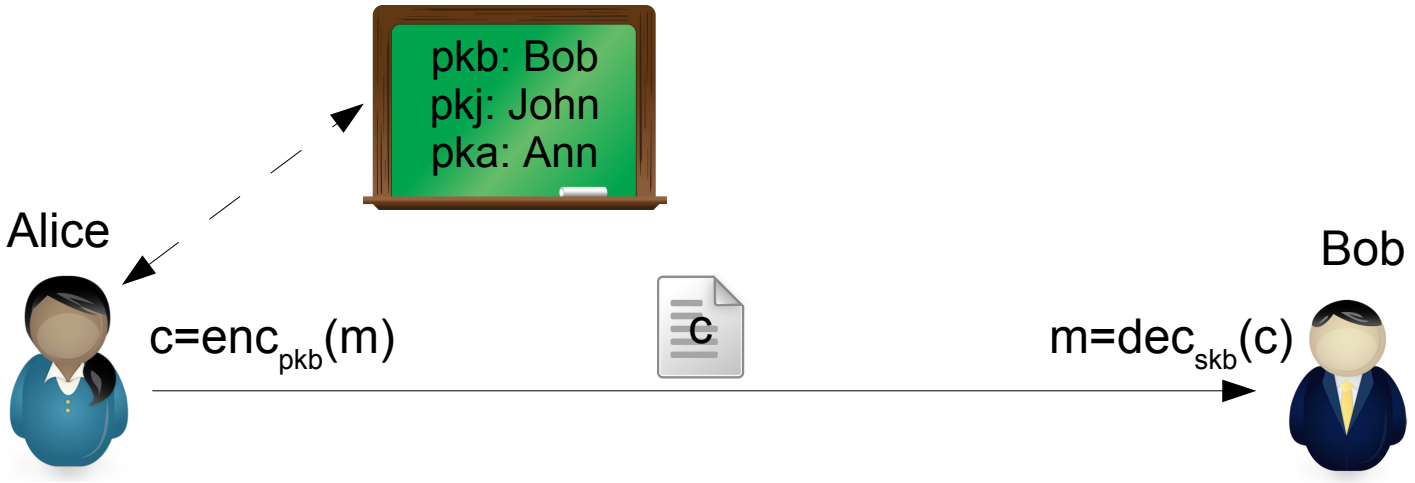
Themen zur Computersicherheit

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) B 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
  - Rivest-Shamir-Adleman (RSA)
  - 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
  - ϕ(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 $a < p$
    - (Q): p,q prim: $a^{k\phi(p\cdot q)+1} = a \bmod (p\cdot q)$ 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

- seinen p, q Primzahlen
- $n = p \bullet q$
- $\phi(n) = (p-1) \bullet (q-1)$ die Zahlen in $\mathbf{Z}/n\mathbf{Z}$ die zu p und q teilerfremd sind
- (n,e) ist der öffentlicher Schlüssel, wenn e teilerfremd zu $\phi(n)$
- (p,q,d) mit $d = e^{-1} \bmod \phi(n)$ ist der dazugehörige private Schlüssel
  - $\Rightarrow e \bullet d = 1 + k \bullet \phi(n)$
- 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 (Q) aus Satz von Euler:
      - (Q): p,q prim: $a^{k\phi(p \bullet q)+1} = a \bmod (p \bullet q)$ für alle a

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 genau ein x∈**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 ist
    z.B. Langzahlmultiplikation
    - naiv: $O(n^2)$
    - Karazuba $O(n^{\mathrm{ld}\ 3})$
    - Schönhage-Strassen $O(n \cdot (\log n) \cdot (\log (\log n)))$
- RSA: der private Schlüssel wir oft im CRT Format angegeben:
  - (p, q, d_p, d_q, q^{-1}) mit $d = d_p \bmod p$ und $d = d_q \bmod q$

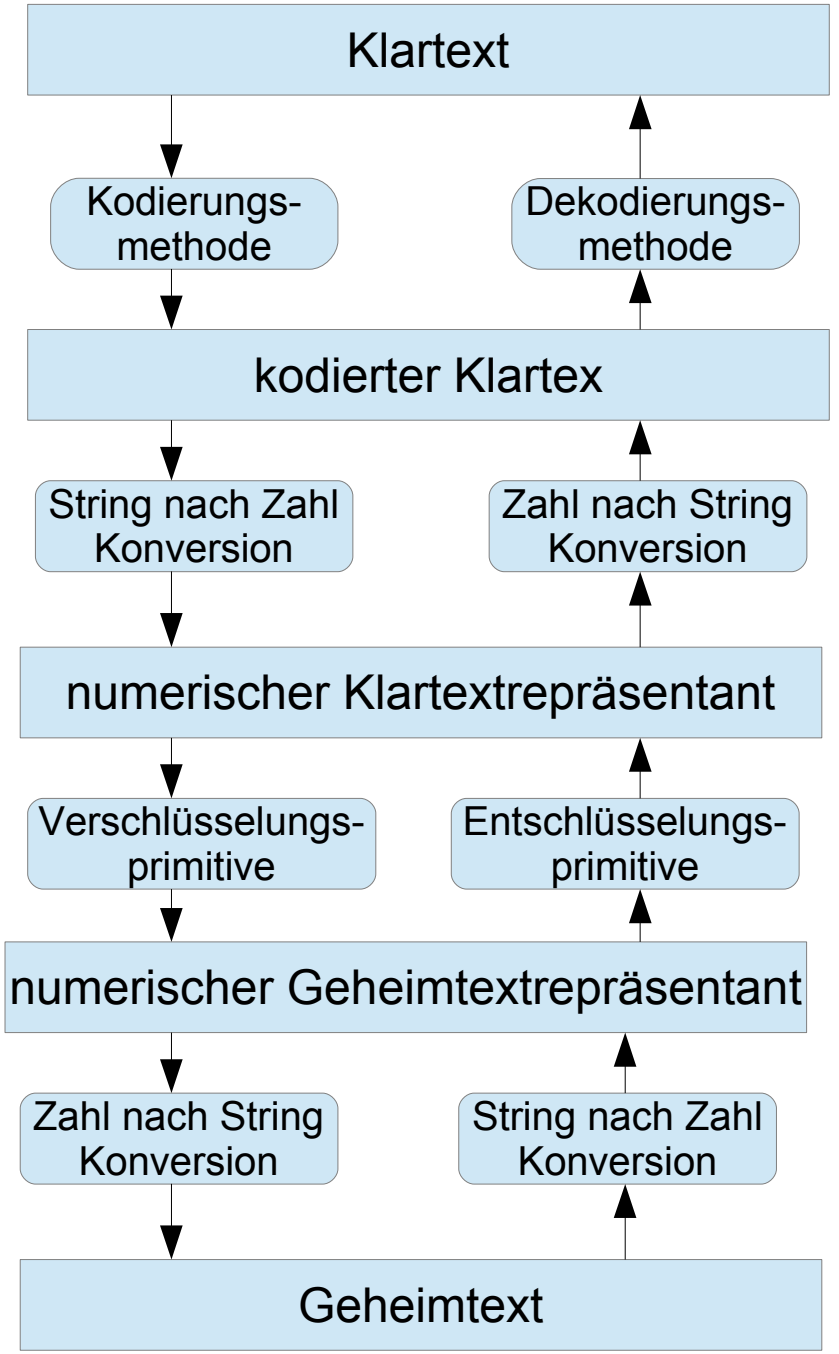
Einschränkungen und Fallstricke

- Der numerische Wert der Nachricht INT(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.
  - 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

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



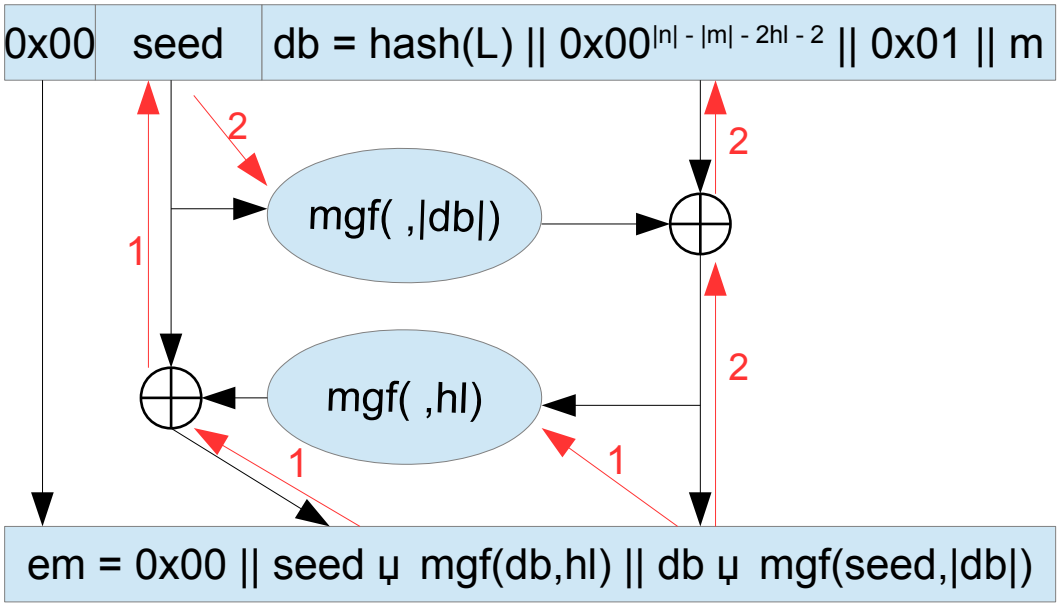
PKCS#1 Kodierungsoperationen

PKCS1-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 erstem 0x00 Byte nach dem zweiten Byte

PKCS1-OAEP Kodierung

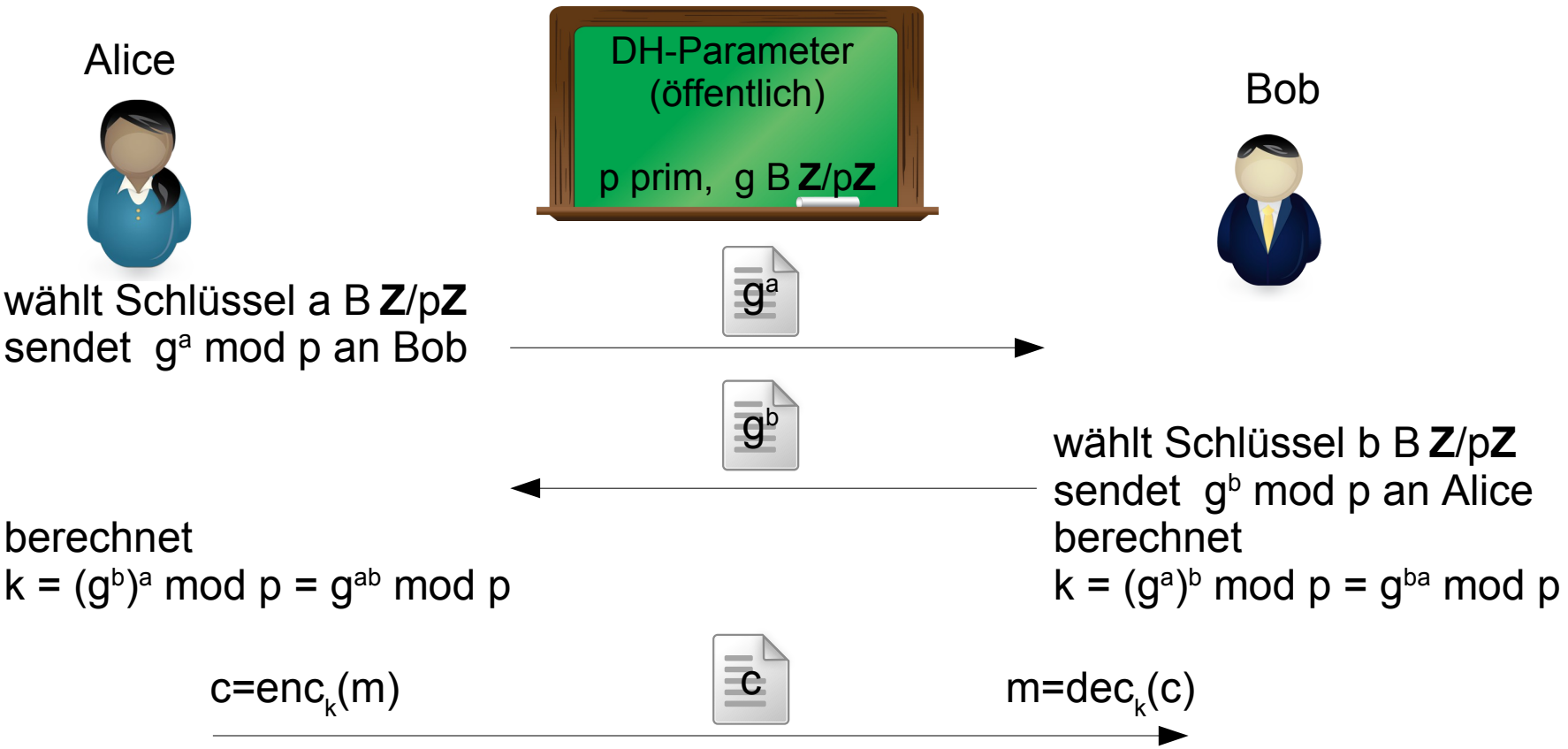
- Eingabe
  - Länge des Modulus: |n| in Bytes
  - Hashfunktion: hash: $\sum \rightarrow \sum^{hl}$
  - maskenerzeugende Funktion: mgf: $\sum^* \times \mathbf{N} \rightarrow \sum^*$
  - optionales Label: L B $\sum^*$
  - seed: Zufallsbytestring der Länge hl



- 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 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 B **Z**/p**Z** öffentlich



Ein paar Begriffe aus der Gruppentheorie

- p prim, dann ist $Z_p^* = \{1, \ldots, p-1\}$ die multiplikative Gruppe modulo p
- für a B 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^*

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 kryptographischen Hashfunktion von k abgeleitet.

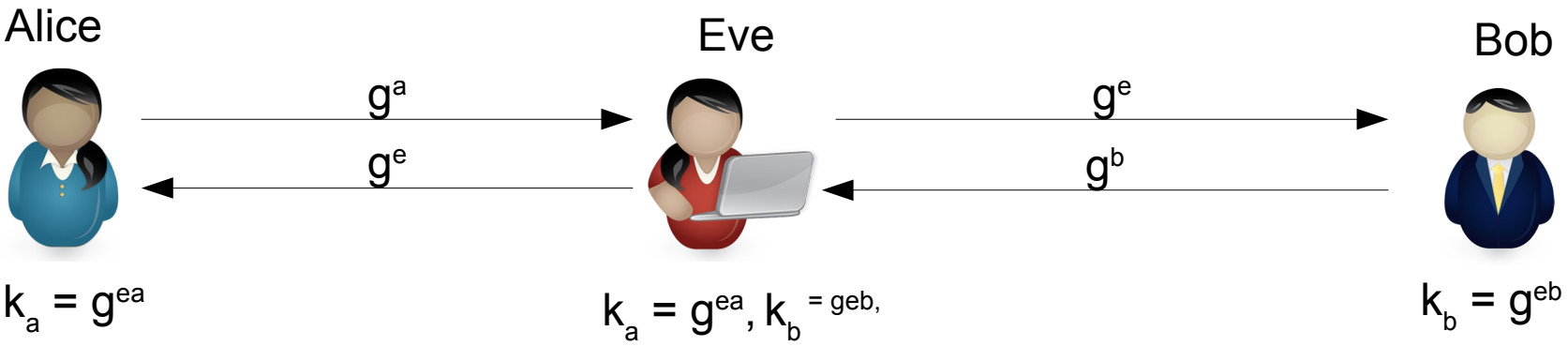
Die Wahl des DH-Parameters g

- { 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
    - => für α B {2, …, p-2} mit $\alpha^2 \neq 1$ mod p und $\alpha^2 \neq p-1$, setze g = α^2
    - (es gibt einen einfachen Test um festzustellen ob h mod p eine Quadratzahl ist)

- Effizienter Schlüsselaustausch
  - nutze g, das kleinere Untergruppe der Ordnung q erzeugt
  - dann gilt: g^x mod p = $g^{x \bmod q}$ mod 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 = α^N mod p, g ≠ 1 mod p, g ≠ p-1 mod 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)

Varianten von DH

- Elliptic Curve Diffie-Hellman (ECDH)
- Elliptische Kurve über endlichen Körper GF(p^n): { (x,y) B GF(p^n)^2 | $y^2 = x^3 + ax + b$ }
- auf Elliptischen Kurven lässt sich eine (multiplikative) Gruppe beschreiben mit
  - $(x_1,y_1) \bullet (x_2, y_2) = (x_3, y_3)$ 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: finde n so das $x = a^n$
  - das Problem des diskreten Logarithmus ist schwerer als bei Standard-DH → kleiner Modulus