

Konzepte von Betriebssystemkomponenten:  
Schwerpunkt Sicherheit

Thema:

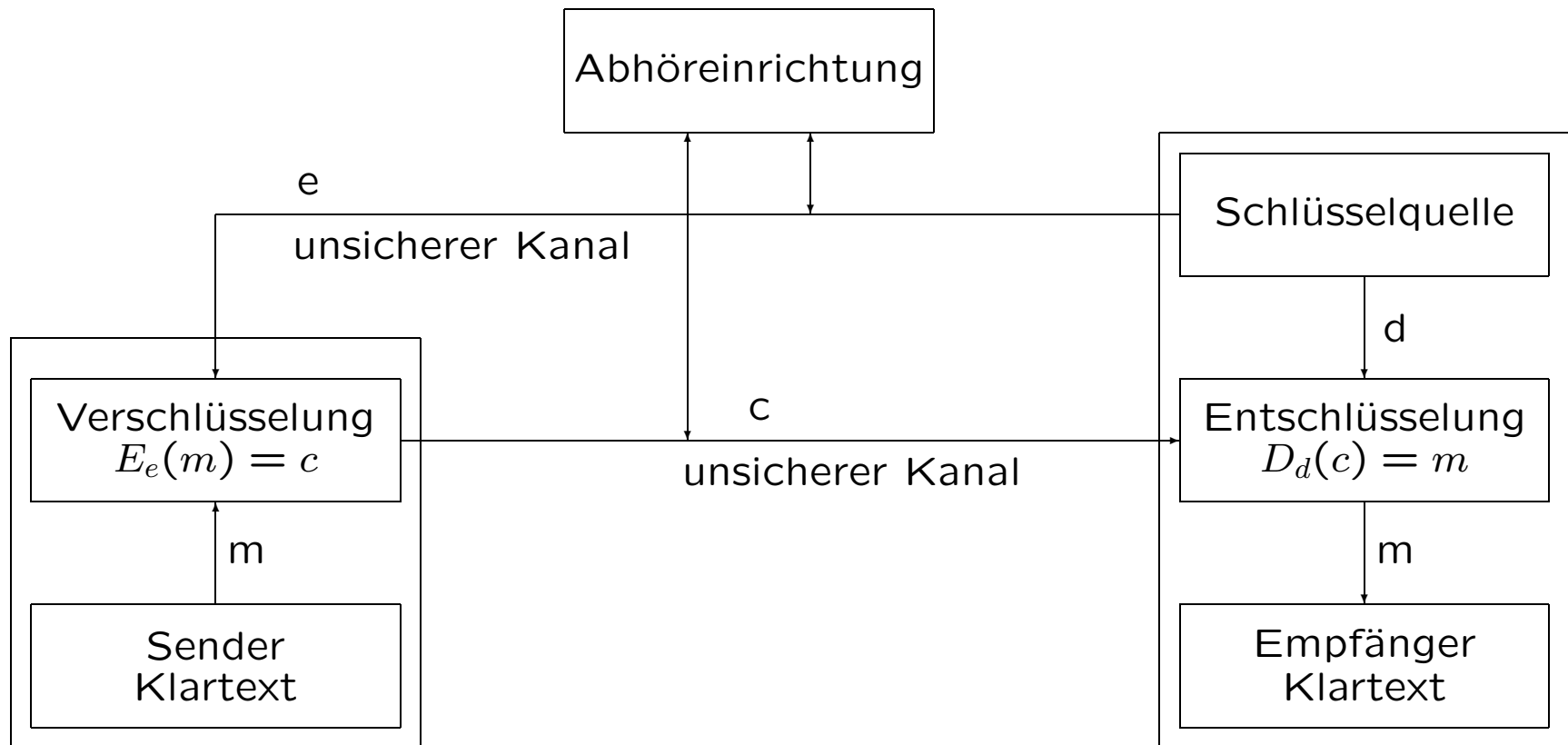
Asymmetrische Verschlüsselung, Digitale Signatur

Vortragender:  
Rudi Pfister

## Überblick:

- Asymmetrische Verschlüsselungsverfahren - Prinzip
- RSA - Beispiel für ein asymmetrisches Verschlüsselungsverfahren
- Weitere Beispiele für Public-Key-Verfahren
- Vor- und Nachteile der asymmetrischen Verschlüsselung
- Digitale Signaturen

# Prinzip der asymmetrischen Verschlüsselung



c: verschlüsselte Nachricht

m: Nachricht in Klartext

e: öffentlicher Schlüssel

d: privater Schlüssel

## Forderungen an Verschlüsselungsmechanismen

- $D_i \circ E_i = \text{ident}$
- Effiziente Realisierungen für  $E_i$  und  $D_i$
- Globale Unberechenbarkeit:  $d_i$  darf praktisch nicht aus  $e_i$  ermittelt werden können.
- Lokale Unberechenbarkeit:  $m$  darf praktisch nicht aus  $c$  ohne Kenntnis von  $d_i$  berechnet werden können.

## Praktische Sicherheit

- Die Umkehrung der Chiffrieroperation oder das Ableiten des Dechiffrierschlüssels scheitern nicht aus prinzipiellen Gründen, sondern einzig und allein am dafür erforderlichen Aufwand.
- Folge: Nötige Parameter und Schlüssel müssen entsprechend gewählt werden.  
⇒ Schlüssel- und Zahlenlänge

## Probleme und offene Fragen

- Authentizität der Schlüssel
- Da Schlüssel öffentlich, kann Verschlüsselung schon vor der ersten Benutzung geknackt werden.
- Wahl geeigneter Funktionen, die die Forderungen erfüllen.

## Verschlüsselung mit RSA

Beispiel:

Teilnehmer A möchte Teilnehmer B eine verschlüsselte Nachricht senden

## Schlüsselerzeugung (Teilnehmer B)

- Wahl zweier geeigneter Primzahlen  $p$  und  $q$  ( $p$  ungleich  $q$ )
- Berechnung von  $n = p * q$  und  $\Phi(n) = (p - 1)(q - 1)$
- Wahl einer natürlichen Zahl  $e$  mit  $1 < e < \Phi(n)$  und  $\text{ggT}(e, \Phi(n)) = 1$
- Berechnung von  $d = e^{-1} \text{ modulo } \Phi(n)$  ( $d$  ist privater Schlüssel)
- Bekanntmachen von  $e$  und  $n$  als öffentlichen Schlüssel
- $d, p, q$  und  $\Phi(n)$  müssen geheim bleiben !

## Verschlüsseln (Teilnehmer A)

- Umwandeln des Klartextes in Zahlenfolge  $m_1, m_2, \dots$  mit  $0 \leq m_i \leq n - 1$
- Berechnen von  $c_i = m_i^e \text{ modulo } n$
- Senden der verschlüsselten Nachricht  $c_1, c_2, \dots$  an Teilnehmer A

## Entschlüsseln (Teilnehmer B)

- Berechnen von  $m_i = c_i^d \text{ modulo } n$
- Umwandeln der Zahlenfolge  $m_1, m_2, \dots$  in Klartext

## Anmerkungen zur Sicherheit von RSA

- Könnte ein Angreifer  $n$  faktorisieren, so könnte er den geheimen Schlüssel  $d$  sehr schnell berechnen.
  - ⇒  $p$  und  $q$  müssen gross genug gewählt werden
  - ⇒  $p$  und  $q$  sollten etwa gleiche Bitlänge haben
  - ⇒ Differenz  $p - q$  sollte nicht zu klein sein, da sonst  $p \approx q$  und somit  $p \approx \sqrt{n}$
  - ⇒  $p$  und  $q$  sollten starke Primzahlen sein
- Kleine Nachrichten sind leicht durch Verschlüsseln sämtlicher möglicher Klartexte zu knacken.
  - ⇒ „Versalzen“ der Nachrichten, d.h. anhängen von Zufallszahlen an die zu verschlüsselnde Nachricht.
- Iteration, d.h. wiederholtes Verschlüsseln der Nachricht bis Klartext vorliegt
- Gefahr durch kleines  $e$  (3 in der Praxis üblich), besser z.B.  $2^{16} + 1 = 65537$

## Rabin

- Beruht auf dem Problem Quadratwurzeln modulo  $n$  zu finden. Sicherheit mathematisch bewiesen.
- Privater Schlüssel: Primzahlen  $p$  und  $q$ , beide kongruent 3 modulo 4
- öffentlicher Schlüssel:  $n = p * q$
- Verschlüsseln  
 $c = m^2 \text{ modulo } n$

- Entschlüsseln

$$t_1 = c^{(p+1)/4} \text{ modulo } p$$

$$t_2 = (p - c^{(p+1)/4}) \text{ modulo } p$$

$$t_3 = c^{(p+1)/4} \text{ modulo } q$$

$$t_4 = (p - c^{(p+1)/4}) \text{ modulo } q$$

$$a = q(q^{-1} \text{ modulo } p)$$

$$b = p(p^{-1} \text{ modulo } q)$$

$$m_1 = (at_1 + bt_3) \text{ modulo } n$$

$$m_2 = (at_1 + bt_4) \text{ modulo } n$$

$$m_3 = (at_2 + bt_3) \text{ modulo } n$$

$$m_4 = (at_2 + bt_4) \text{ modulo } n$$

Eine der Lösungen  $m_1, \dots, m_4$  ist  $m$

⇒ Verfahren zur Wahl der Richtigen Lösung nötig

## ElGamal

- Beruht auf der Schwierigkeit diskrete Logarithmen in endlichen Körpern zu Berechnen
- Privater Schlüssel:  $x$
- Öffentlicher Schlüssel:  $p$ ,  $g$  und  $y = g^x \text{ modulo } p$
- Verschlüsselung  
k Zufallszahl  
 $a = g^k \text{ modulo } p$   
 $b = y^k m \text{ modulo } p$
- Entschlüsseln  
 $m = b/a^x \text{ modulo } p$
- Nachteil: Verschlüsselter Text doppelt so gross wie Klartext

## McEliece

- Basiert auf fehlerkorrigierenden Codes, sog. **Goppa Codes**
- Verfahren zwei- bis dreimal so schnell wie RSA
- Sehr lange Schlüssellänge, deswegen kaum verwendet

## Merkle-Hellmann-Untersummen

- Zugrundeliegendes NP-vollständiges Problem:
  - n Pakete mit unterschiedlichen Gewichten  $M_1, M_2, \dots, M_n$  und vorgegebenes Gesamtgewicht  $S$ .
  - ist  $S = b_1M_1 + b_2M_2 + \dots + b_nM_n$  mit  $b_i = \{0, 1\}$  möglich ?
- Superincreasing-Knapsack-Problem
  - $M_i > M_1 + M_2 + \dots + M_{i-1}$  für  $i = 2, 3, \dots, n$
  - lineare Komplexität, keine Sicherheit
- Merkle und Hellman transformieren Superincreasing-Knapsack-Problem in allgemeines Knapsack-Problem mit Transformationsparametern als Schlüssel.
- Die Unsicherheit des Verfahrens wurde nachgewiesen.

## Vor- und Nachteile Public-Key-Verschlüsselung

- + Für jeden Teilnehmer ist nur ein Schlüsselpaar nötig.
- + Es ist spontane gesicherte Kommunikation möglich - ohne vorherigen Schlüsseltausch über einen sicheren Kanal.
  - Die Realisierungen in Hard- und Software sind im allgemeinen aufwendiger als symmetrische Verfahren.
  - Die Durchsatzraten liegen deutlich unter den mit vergleichbaren symmetrischen Verfahren. Der Haupteinsatz der asymmetrischen Verfahren liegt deshalb in der chiffrierten Übertragung eines Schlüssels für ein symmetrisches Verfahren.
  - Die Sicherheit der wichtigsten asymmetrischen Verfahren hängt an der vermuteten aber bislang unbewiesenen Schwierigkeit gewisser zahlentheoretischer Probleme (Faktorisierung, diskreter Logarithmus).

## Digitale Signaturen

- Digitale Signaturen sollen die Identität des Absenders und die Integrität der übertragenen Daten belegen.
- Zwei Prinzipien:
  - Giving Message Recovery
  - With Appendix

## Signieren mit DSA

Beispiel:

Teilnehmer B möchte die Nachrichten von A auf Autentität prüfen

## Schlüsselerzeugung (Teilnehmer A)

- Wahl von Primzahl  $p$
- Wahl von  $q$  mit  $q$  ist Primteiler von  $(p - 1)$
- Berechnen von  $g = h^{(p-1)/q} \text{ modulo } p$  mit  $h < p-1$  und  $h^{(p-1)/q} \text{ modulo } p > 1$
- $g, p$  und  $q$  sind öffentlich und können von mehreren Teilnehmern genutzt werden
- Wahl von Zufallszahl  $x$  mit  $0 < x < q$  als privaten Schlüssel
- Berechnen von  $y = g^x \text{ modulo } p$  als öffentlichen Schlüssel

## Signieren einer Nachricht $m$ (Teilnehmer A)

- Wahl von Zufallszahl  $k$  mit  $0 < k < q$
- Berechnen von  $r = (g^k \text{ modulo } p) \text{ modulo } q$
- Berechnen von  $s = (k^{-1}(H(m) + xr)) \text{ modulo } q$ , wobei  $H$  eine Ein-Weg-Hash-Funktion ist
- $r$  und  $s$  bilden die Signatur, die zusammen mit der Nachricht  $m$  übermittelt werden.

## Verifizieren der Nachricht (Teilnehmer B)

- Berechnen von  $w = s^{-1} \text{ modulo } q$
- Berechnen von  $u_1 = (H(m) * w) \text{ modulo } q$
- Berechnen von  $u_2 = (rw) \text{ modulo } q$
- Berechnen von  $v = ((g^{u_1} * y^{u_2}) \text{ modulo } p) \text{ modulo } q$
- Wenn  $v = r$ , dann ist die Nachricht verifiziert

## Weitere Verfahren für Digitale Signaturen

- GOST
- Ong-Schnorr-Shamir
- ESIGN
- Asymmetrische Verschlüsselungsverfahren zum Signieren

## Sicherheit der Signaturen

- Ähnlich wie Public-Key-Verschlüsselung, da ähnliche oder sogar gleiche Algorithmen.