

# Public Key Kryptographie

Prof. Dr. Rüdiger Weis

TFH Berlin

Sommersemester2008

# Algorithmen für Langzahlen

RSA

Das Rabin-Kryptosystem

Diskrete Logarithmen

**Bisher:** Ein Schlüssel für Sender **und** Empfänger

*('Secret-Key' oder "symmetrische" Kryptographie*

**Nun:** Ein Teil des Schlüssels ist nur dem Empfänger bekannt. Der auch dem Sender bekannte Teil kann sogar „veröffentlicht“ werden. Man spricht dann von einem **Schlüsselpaar**, bestehend aus

- ▶ dem 'Public Key' für den Sender und
- ▶ dem 'Private Key', exklusiv für den Empfänger. Ohne Private Key kann nicht entschlüsselt werden!

*'Public Key' oder "asymmetrische" Kryptographie)*

Alice sendet eine Nachricht an Bob.

Beide verwenden symmetrische Kryptographie („secret-key“).

- ▶ **Problem:** Schlüsseltransport  
**Lösung:** Alice schickt den Schlüssel an Bob ???
- ▶ **Problem:**  $T$  Teilnehmer,  $T$  *sehr groß*  
**Lösung:** Speichern von  $T(T - 1)/2$  Schlüsseln ???

# Public-Key Kryptographie (Beispiele)

- ▶ Briefkasten
- ▶ „No-Key Protokoll“ von Shamir
- ▶ Asymmetrische Kryptographie („public-key“)  
*Schloß mit zwei Schlüsseln*
- ▶ Einwegfunktionen mit Falltür

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

**1974** „Merkle Puzzles“

**1976** Diffie und Hellman

**1977** Rivest, Shamir, Adleman (RSA)

**seit etwa 1990** zunehmende kommerzielle Bedeutung der asymmetrischen Kryptographie

# Zur Erinnerung (1)

$$\begin{aligned}a|b &\Leftrightarrow \text{es gibt } k \text{ mit } k * a = b \\ \mathbb{Z}_n &= \{0, 1, \dots, n-1\} \quad (*) \\ a \equiv b \pmod{n} &\Leftrightarrow n|(a-b) \\ \mathbb{Z}_n^* &= \{i \in \mathbb{Z}_n \mid \text{ggT}(i, n) = 1\} \quad (*)\end{aligned}$$

(\*): Kanonische Darstellung

- ▶  $(\mathbb{Z}_n, +)$  ist eine Gruppe.
- ▶  $(\mathbb{Z}_n^*, *)$  ist eine Gruppe.

## Zur Erinnerung (2)

Euler'sche  $\varphi$ -Funktion:  $\varphi(n) = |\mathbb{Z}_n^*|$

Beispiele für  $\mathbb{Z}_n$ ,  $\mathbb{Z}_n^*$  und  $\varphi(n)$ :

$n$	$\mathbb{Z}_n$	$\mathbb{Z}_n^*$	$\varphi(n)$
2	$\mathbb{Z}_2 = \{0, 1\}$	$\mathbb{Z}_2^* = \{1\}$	$\varphi(2) = 1$
3	$\mathbb{Z}_3 = \{0, 1, 2\}$	$\mathbb{Z}_3^* = \{1, 2\}$	$\varphi(3) = 2$
4	$\mathbb{Z}_4 = \{0, 1, 2, 3\}$	$\mathbb{Z}_4^* = \{1, 3\}$	$\varphi(4) = 2$
5	$\mathbb{Z}_5 = \{0, \dots, 4\}$	$\mathbb{Z}_5^* = \{1, 2, 3, 4\}$	$\varphi(5) = 4$
6	$\mathbb{Z}_6 = \{0, \dots, 5\}$	$\mathbb{Z}_6^* = \{1, 5\}$	$\varphi(6) = 2$
7	$\mathbb{Z}_7 = \{0, \dots, 6\}$	$\mathbb{Z}_7^* = \{1, \dots, 6\}$	$\varphi(7) = 6$
8	$\mathbb{Z}_8 = \{0, \dots, 7\}$	$\mathbb{Z}_8^* = \{1, 3, 5, 7\}$	$\varphi(8) = 4$



# Zwei wichtige Eigenschaften von $\varphi$

## Theorem

- ▶  $p$  ist prim  $\Leftrightarrow \varphi(p) = p - 1$ .
- ▶  $p \neq q$  sind beide prim  
 $\Rightarrow \varphi(p \cdot q) = \varphi(p) \cdot \varphi(q) = (p - 1) \cdot (q - 1)$

## Theorem (Satz von Euler)

*Ist  $\text{ggT}(a, n) = 1$ , so gilt*

$$a^{\varphi(n)} \equiv 1 \pmod{n}.$$

(Die Beweise sind einfach und in einführenden Lehrbüchern über Zahlentheorie zu finden.)

# Algorithmen für (große) ganze Zahlen

- ▶ Grundrechenarten (+, −, \*, div, mod)
- ▶ Modulares Potenzieren:  $a^b \bmod m$  (“square-and-multiply”)
- ▶ ggT und Inverse in  $\mathbb{Z}_n^*$   
(Euklidischer und erweiterter Euklidischer Alg.)
- ▶ Chinesischer Restsatz (\*)
- ▶ zusammengesetzte Zahlen faktorisieren (\*)
- ▶ Primzahlen testen/finden (\*)

## Theorem (Chinesischer Restsatz)

Seien  $m_1, \dots, m_k$ : paarweise teilerfremde natürliche Zahlen, mit  $m = m_1 * \dots * m_k$  und  $a_1, \dots, a_k$ : ganze Zahlen.

Für  $1 \leq i \leq k$  gelte  $M_i = m/m_i$  und  $y_i = M_i^{-1} \pmod{m_i}$ .

Dann gilt für

$$x \equiv \left( \sum_i a_i y_i M_i \right) \pmod{m} :$$

$x \equiv a_1 \pmod{m_1}$ ,  $x \equiv a_2 \pmod{m_2}$ ,  $\dots$ , und  $x \equiv a_k \pmod{m_k}$ .

# Spezialfall des Chinesischen Restsatzes

## Theorem

Sei  $m = m_1 * m_2$  mit  $\text{ggT}(m_1, m_2) = 1$ ;

sei  $y_1 = m_2^{-1} \text{ mod } m_1$ ,

und sei  $y_2 = m_1^{-1} \text{ mod } m_2$ .

Für  $a_1, a_2 \in \mathbb{Z}$  und  $x = a_1 y_1 m_2 + a_2 y_2 m_1$  gilt:

$$x \equiv a_1 \text{ mod } m_1 \text{ und } x \equiv a_2 \text{ mod } m_2.$$

# Wozu brauchen wir den Chinesischen Restsatz?

- ▶ Teilerfremde Zahlen  $p, q$ ; Produkt  $n = pq$ .
- ▶ Gegeben  $x_p, x_q$ .
- ▶ Gesucht:  $x \in \mathbb{Z}_n$ :

$$x \equiv x_p \pmod{p}$$

und

$$x \equiv x_q \pmod{q}.$$

Man berechne  $x$  mit Hilfe des Chinesischen Restsatzes!

**Aufgabe:** Gegeben Werte  $a$  und  $B$  mit  $a < B - 1$ . Finde eine zufällige Primzahl  $p$  mit  $p \geq a$  und  $p < B$ .  
(Typisch:  $a = 2^n$ ,  $B = 2a = 2^{n+1}$ .)

**Lösung:**

Wiederhole:

- ▶ wähle eine Zufallszahl  $z \in \{a, \dots, B - 1\}$ ,
- ▶ teste, ob  $z$  prim ist oder zusammengesetzt,

bis  $z$  eine Primzahl ist.

Gib  $z$  aus.

## Primzahlen finden(2)

**Frage:** Wie effizient ist dieses Verfahren?

- ▶ Wie oft wird die Schleife durchlaufen?

*(Häufigkeit der Primzahlen) (\*)*

- ▶ Kann man effizient testen, ob  $z$  eine Primzahl ist?

*Es gibt sehr effiziente probabilistische Algorithmen für Primzahltests (z.B. „Miller-Rabin“).*

*Vor kurzem fanden indische Informatiker sogar einen deterministischen Primzahltest in Polynomialzeit. Dies ist ein interessantes theoretisches Ergebnis („PRIMES  $\in$  P“).*



# Häufigkeit der Primzahlen

Def.:  $\pi(x)$  ist die Anzahl der Primzahlen  $\leq x$ .

Bsp.:  $\pi(1) = 0$ ,  $\pi(3) = 2 = \pi(4) = 2$ ,  $\dots$ ,  $\pi(124) = 30$ .

Dem *Primzahlsatz* zufolge gilt

$$\pi(x) \approx \frac{x}{\ln(x)}.$$

Diese Approximation ist sogar recht genau. Für  $x \geq 17$  gilt z.B.

$$\frac{x}{\ln(x)} < \pi(x) < 1,25506 \frac{x}{\ln(x)}.$$

# Faktorisierung großer ganzer Zahlen

Das **Faktorisieren** großer ganzer Zahlen gilt als extrem schwierige Aufgabe. Dies ist etwas überraschend, da doch die **Multiplikation** einfach ist – sogar der **Test, ob eine Zahl prim ist**, ist ja vergleichsweise einfach.

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

*Obwohl er die Sonntage dreier Jahre benötigte, um die Faktoren von  $2^{67} - 1$  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 von  $2^{67} - 1$  und  $193707721 * 761838257287$  aufschrieb.*

- Zwei zufällig gewählte *große* Primzahlen  $p$  und  $q$ .  
 Seien  $n = pq$  und  $e \in \mathbb{Z}_{\varphi(n)}^*$ , d.h.,  $\text{ggT}(e, \varphi(n)) = 1$ .  
 Sei  $d$  das multiplikative Inverse von  $e$  modulo  $\varphi(n)$
- Klartextmenge = Chiffretextmenge =  $\mathbb{Z}_n$
- Schlüssel ist das Tripel  $(e, d, n)$ .  
 Öffentlicher Schlüssel ist das Paar  $(e, n)$ .
- Verschlüsselungsfunktion  $E$ :  $E_{(e,n)}(x) = x^e \bmod n$
- Entschlüsselungsfunktion  $D$ :  $D_{(e,d,n)}(y) = y^d \bmod n$

## Beispiel mit kleinen Zahlen

Beispiel:  $p = 13$ ,  $q = 11$ ,  $n = pq = 143$ .

Es ist  $\varphi(n) = (p - 1)(q - 1) = 120$ .

Wir wählen  $e = 7$ .

Insbesondere ist  $\text{ggT}(7, 120) = 1$ .

Für  $d = 103$  gilt dann:  $ed = 721 = 6 * 120 + 1 \equiv 1 \pmod{120}$ .

Wenn wir den Klartext 5 Verschlüsseln, erhalten wir

$$E(5) \equiv 5^7 \equiv 78125 \equiv 47 \pmod{143}.$$

Wenn wir 47 wieder entschlüsseln, gilt:

$$D(47) \equiv 47^{103} \equiv 5 \pmod{143}.$$

# Sicherheit und Korrektheit 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).

## Theorem (Korrektheit von RSA (1))

Für  $x \in \mathbb{Z}_n^*$  gilt:  $D(E(x)) = x$ .

Im Falle des RSA-Systems gilt zusätzlich:

## Theorem (Korrektheit von RSA (2))

Für  $x \in \mathbb{Z}_n^*$  gilt:  $E(D(x)) = x$ .

# „Vorstellbare Angriffe“ auf RSA

1. Faktorisieren von  $n$ . (Gilt als extrem schwierig!)
2. Berechnen von  $\varphi(n)$ .  
(Kann man das, kann man auch Faktorisieren.)
3. Ermitteln eines  $d'$  mit  $x^{ed'} \equiv x^{ed} \equiv x \pmod{n}$ . Man beachte, dass  $d = d'$  nicht gelten muss.  
(Kann man das, kann man auch Faktorisieren. Der Beweis ist allerdings nicht gar so einfach. Er beruht darauf, daß  $ed' = k\varphi(n) + 1$  ist und man auf  $\varphi(n)$  zurückschließen kann. )

## „Vorstellbare Angriffe“ (2)

4. Berechnen der  $e$ -ten Wurzel modulo  $n$  („RSA-Wurzel“).  
(*Vermutlich* ebenso schwierig wie die Faktorisierung.)
5. Iteriertes Verschlüsseln des Kryptogramms.  
(Erfordert mit überwältigender Wahrscheinlichkeit astronomisch viele Iterationen.)



# Iteriertes Verschlüsseln

Geg.  $y \in \mathbb{Z}_m^*$ , wende den folgenden Algorithmus an:

- ▶  $x := y$ .
- ▶ Solange  $x^e \not\equiv y \pmod{n}$ :  $x := x^e \pmod{n}$ .
- ▶ Gib  $x$  aus. (\* Nun gilt offenbar  $y \equiv x^e \pmod{n}$ . \*)

Das Verfahren terminiert nach endlicher Zeit. (Warum?)

In der Praxis ist die *endliche Zeit* (also die Anzahl der Iterationen) fast immer astronomisch groß.

# Beispiel für iteriertes Verschlüsseln

$e = 17$ ,  $d = 157$ ,  $n = 2773$ .

Die Chiffretext  $y = 2209$  sei gegeben. Was ist der zugehörige Klartext  $x$  mit  $E(x) = y$ ?

$$E(2209) = 1504$$

$$E(1504) = 2444$$

$$E(2444) = 470$$

$$E(470) = 2209$$

Heureka! Es ist  $x = 470$ .

# Das Rabin-Kryptosystem

Wie RSA, aber  $e = 2$ .

Wenn man Rabin effizient „knacken“ kann,  
dann kann man auch effizient faktorisieren.

# Rabin: Eine Variante von RSA

## Algorithmische Grundlage:

- ▶ Für zusammengesetztes  $n$  ist es extrem schwierig, die Gleichung  $x^2 = y \pmod n$  zu lösen.
- ▶ Es gibt dagegen einen Algorithmus die Gleichungen

$$x_p^2 \equiv y \pmod p \quad \text{und} \quad x_q^2 \equiv y \pmod q$$

löst, wenn  $p$  und  $q$  prim sind.

- ▶ Kennt man zwei derartige Lösungen  $x_p$  und  $x_q$ , kann man den Chinesischen Restsatz benutzen, um die folgende Gleichung zu lösen:

$$x^2 \equiv y \pmod{pq}.$$

# Rabin (Grundlagen)

Für eine ungerade Primzahl  $p$  gilt:

Sei  $y \in \mathbb{Z}_p^*$ . Die Gleichung  $y = x^2 \pmod{p}$  hat entweder gar keine oder genau zwei Lösungen:

Ist  $y \equiv x^2 \pmod{p}$ , dann ist auch  $y \equiv (-x)^2 \pmod{p}$ .

(Beachte:  $-x \equiv p - x \pmod{p}$ .)

# Rabin (weitere Grundlagen)

Für ungerade Primzahlen  $p \neq q$  und das  $n = pq$  gilt:

Sei  $y \in \mathbb{Z}_n^*$ . Die Gleichung  $y = x^2 \pmod n$  hat entweder keine oder genau vier Lösungen:

1. Eine Gleichung  $y \equiv x^2 \pmod p$  (\*) bzw.  $y \equiv x^2 \pmod q$  (\*\*) hat entweder keine oder genau zwei Lösungen.
2. Hat (mindestens) eine der Gleichungen (\*) und (\*\*) keine Lösung, dann hat die Gleichung  $y = x^2 \pmod n$  auch keine Lösung.
3. Sonst ergibt jede Kombination von Lösungen mod  $p$  und mod  $q$  mit Hilfe des Chinesischen Restsatzes eine Lösung mod  $n$ .

# Rabin (Definition)

1. Seien  $p$ ,  $q$  und  $n$  wie beim RSA-System definiert.  
Das Produkt  $n = pq$  dient als öffentlicher Schlüssel, die Primfaktoren  $p$  und  $q$  sind geheim.
2. Verschlüsselungsfunktion  $E$ :

$$E[x] = x^2 \bmod n.$$

3. Entschlüsselungsoperation  $D$ ???

# Rabin Entschlüsselung

- ▶ Berechne die beiden Lösungen der Gleichung  $y \equiv x^2 \pmod{p}$ .
- ▶ Berechne die beiden Lösungen der Gleichung  $y \equiv x^2 \pmod{q}$ .
- ▶ Berechne daraus die vier Lösungen der Gleichung  $y \equiv x^2 \pmod{n}$ .
- ▶ Entscheide, welche der vier Lösungen „richtig“ ist. (Dies kann man z.B. erreichen, indem man die Klartexte mit einer „Prüfsumme“ verknüpft.)



**Strenggenommen ist  $D$  gar keine Funktion!**

Was tun?

Redundanter Klartext  $x$ , der von den „falschen Brüdern“ mit hoher Wahrscheinlichkeit unterschieden werden kann.

# Rabin (Sicherheitsbeweis)

## Theorem

*Seien  $n = pq$  ein Rabin-Modulus und  $A$  ein Algorithmus zur Berechnung von Quadratwurzeln modulo  $n$ . Der folgende Algorithmus liefert mit mindestens der Wahrscheinlichkeit 0.5 einen Primfaktor von  $n$ :*

1. *Wähle zufällig  $r \in \mathbb{Z}_n$ .*
2. *Berechne  $y = r^2 \bmod n$ .*
3. *Berechne  $t = \text{ggT}(y, n)$ . Wenn  $t > 1$ , gib  $t$  aus. STOP*
4. *Rufe  $A$  auf zur Berechnung von  $x$  mit  $x^2 \equiv y \pmod n$ .*
5. *Ist  $x \equiv \pm r$ , gib 1 aus. Sonst gib  $\text{ggT}(x + r, n)$  aus. STOP*

Aufgabenstellung für ein kryptographisches Protokoll:

Ein Disput zwischen Alice und Bob soll durch einen Münzwurf entschieden werden

(„Kopf“  $\rightarrow$  Alice gewinnt, „Zahl“  $\rightarrow$  Bob gewinnt).

**Kann man so etwas auch über Telefon oder Internet machen?**

(Das ist natürlich leicht, wenn beide Beteiligte sich auf die Ehrlichkeit des Gegenübers verlassen . . . )

# Münzwurf-Protokoll (1. Teil)

1. Alice wählt Primzahlen  $p$  und  $q$  als Geheimnis und schickt  $n = pq$  an Bob.  
(Intuition: Bob gewinnt  $\Leftrightarrow$  Bob findet  $p$  oder  $q$ .)
2. Bob wählt zufällig  $x \in \mathbb{Z}_n$  und schickt  $y = x^2 \bmod n$  an Alice. (Ist  $\text{ggT}(x, n) > 1$  hat Bob bereits gewonnen.)
3. Alice berechnet  $r$  mit  $(*) r^2 \equiv y \bmod n$ .  
Von den 4 möglichen Lösungen der Gleichung  $(*)$  ist  $r$  zufällig gewählt. Alice schickt  $r$  an Bob.

## Münzwurf-Protokoll (2. Teil)

4. Bob testet  $r^2 \equiv y \pmod{n}$ .
5. Ist  $r \not\equiv \pm x \pmod{n}$ , gewinnt Bob:  $\text{ggT}(r + x, n) \in \{p, q\}$ .
6. Kann Bob keinen Faktor  $p$  bzw.  $q$  angeben, tut Alice dies und gewinnt.

## Kann Alice unentdeckt betrügen?

- ▶ Ist  $n$  eine Primzahl, kann Alice im letzten Schritt selbst keine Faktoren  $p$  und  $q$  angeben.
- ▶ Wählt Alice  $n$  als das Produkt mehrerer Primzahlen, *steigen* Bobs Chancen, einen Teiler  $t|n$ ,  $t \notin \{1, n\}$  zu finden.
- ▶ Ist  $r$  keine Quadratwurzel mod  $n$ , wird dies von Bob entdeckt.

## Kann Bob betrügen?

- ▶ Bob kann versuchen,  $n$  zu faktorisieren.
- ▶ Schafft er dies nicht, dann gilt, egal wie  $y$  gewählt ist: Bob kennt nur (höchstens) zwei Quadratwurzeln von  $y$  und hat keinen Einfluß darauf, welche Quadratwurzel  $r$  Alice ihm zurückgibt.

# Diskrete Logarithmen

- ▶ Primzahl  $p$ , Zahlen  $x, g \in \{1, 2, \dots, p - 2\}$
- ▶ Zahl  $B = g^x \bmod p$
- ▶ Das Problem, zu  $B, g, p$  einen Wert  $x$  zu finden mit  $B = g^x \bmod p$  bezeichnet man als  
„Diskreten Logarithmus von  $B$  zur Basis  $g$ “.  
Dieses Problem gilt als extrem schwierig.



# Diffie-Hellman Schlüsselaustausch

Primzahl  $p$  und Generator  $g$  festgelegt.

Alice: wählt  $a$  als geheimen Schlüssel

öffentlicher Schlüssel:  $A = g^a \bmod p$ .

Bob: wählt  $b$  als geheimen Schlüssel

öffentlicher Schlüssel:  $B = g^b \bmod p$ .

Geheimer Sitzungsschlüssel:  $K = g^{ab} \bmod p$ .

Alice und Bob können den Sitzungsschlüssel effizient berechnen.  
(Wie?)

# Das „Diffie-Hellman Problem“

**Gegeben  $A$  und  $B$ , berechne  $K$ .**

Dieses Problem ist gilt als als extrem schwierig.

Wenn man das DL-Problem effizient lösen kann, kann man das DH-Problem auch effizient lösen. (Warum?)

Ob die Umkehrung auch gilt, ist leider unklar.

# ElGamal-Verschlüsselung

Weiterentwicklung des D.-H.-Schlüsselaustausches:

Primzahl  $p$  und Generator  $g$  festgelegt.

Alice: wählt  $a$  als geheimen Schlüssel

öffentlicher Schlüssel:  $A = g^a \bmod p$ .

Bob: will Nachricht  $m \in \mathbb{Z}_p^*$  verschlüsseln

wählt zufällig  $r \in \mathbb{Z}_p^*$

Berechnet  $B = g^r \bmod p$  und  $C = A^r m \bmod p$

Chiffretext:  $(B, C)$ .

Alice: berechnet  $x = p - 1 - a$

Nachricht:  $m' = B^x * C \bmod p$ .

Es gilt  $m' = m$ . (Nachrechnen!)

Historisch wurden D-H Schlüsselaustausch und ElGamal Verschlüsselung über der Gruppe  $(*, \mathbb{Z}_p^*)$  definiert (oder Untergruppen). Man kann aber auch andere endliche Gruppen  $(\circ, G)$  nutzen, wenn gilt:

1. Die Gruppenoperation  $\circ : G \times G \rightarrow G$  ist *effizient berechenbar*.
2. Das Diffie-Hellman Problem in der Gruppe ist *hart*.

Als besonders vielversprechend gelten *additive Punktgruppen von elliptischen Kurven über endlichen Körpern*. Diese werden Rahmen dieser Vorlesung aber nicht weiter betrachtet.

## Danksagungen

- ▶ Nach einer Vorlesung von Stefan Lucks
- ▶ Erstellt mit Freier Software