FH AACHEN
UNIVERSITY OF APPLIED SCIENCES

Skript zur Vorlesung

Kryptologie

Georg Hoever

Fachbereich Elektrotechnik und
Informationstechnik
FH Aachen

Inhaltsverzeichnis

1 Modulo-Rechnung, Gruppen

Motivation:

Welche ganzen Zahlen lassen bei Division durch 6 den Rest 2?

Antwort: 2, 8, 14, 20, ...

Die Reihe kann man in negativer Richtung fortsetzen: −4, −10, −16, ...

Allgemein erhält man als Werte $2 + k \cdot 6$, $k \in \mathbb{Z}$.

Definition:

Zu zwei ganzen Zahlen a, m, $m > 0$, bezeichnet $a \bmod m$ die eindeutige Zahl $r \in \{0, \ldots, m-1\}$, so dass es ein $k \in \mathbb{Z}$ gibt mit $a = r + k \cdot m$.

Zwei Zahlen $a, b \in \mathbb{Z}$ heißen kongruent modulo m ($a \equiv b \bmod m$) genau dann, wenn $a \bmod m = b \bmod m$.

Bemerkungen:

1. Für $a \geq 0$ ist $a \bmod m$ der Rest bei der Division von a durch m.
2. Die Definition ist in der Literatur nicht ganz einheitlich. Manchmal wird mit $a \bmod m$ auch das $r \in \{1, \ldots, m\}$ oder das betragskleinste r mit $a = r + k \cdot m$ für ein $k \in \mathbb{Z}$ bezeichnet.
3. Statt $a \equiv b \bmod m$ wird oft (so auch im Folgenden) $a = b \bmod m$ geschrieben.

   Damit ist „ $\bmod m$“ einerseits ein Rechenoperator (Bsp.: $18 \bmod 5 = 3$), andererseits bezieht es sich auf eine ganze Gleichung oder Gleichungskette (Bsp.: $18 = 13 \bmod 5$).

   Es gilt:

$$a = b \bmod m \quad \Leftrightarrow \quad \exists\, k \in \mathbb{Z} \text{ mit } a = b + k \cdot m.$$

Erinnerung:

Das Zeichen „$\exists x \in M \ldots$“ bedeutet „Es gibt ein $x \in M$, für das gilt ...“.

Das Zeichen „$\forall x \in M \ldots$“ bedeutet „Für alle $x \in M$ gilt ...“.

Beispiele:

$20 \bmod 6 = 2$,
$16 \bmod 4 = 0$,
$-2 \bmod 7 = 5$,
$17 = 12 \bmod 5$,
$2 \cdot 3 = 1 \bmod 5$.

Beobachtung:

$$\begin{array}{cc} 17 \bmod 6 = 5 & -1 \bmod 6 = 5 \\ 14 \bmod 6 = 2 & 8 \bmod 6 = 2 \\ (17+14) \bmod 6 = 1 & (8+(-1)) \bmod 6 = 1 \end{array}$$

Satz 1.1:

Ist $a_1 = a_2 \bmod m$ *und* $b_1 = b_2 \bmod m$, *so gilt*

$$a_1 + b_1 = a_2 + b_2 \bmod m \qquad \textit{und} \qquad a_1 \cdot b_1 = a_2 \cdot b_2 \bmod m.$$

Beispiel:

Es ist

$$17 \cdot 14 = 238 = 4 \bmod 6 \quad \text{und} \quad ((-1)\cdot 8) = -8 = 4 \bmod 6,$$

also $17 \cdot 14 = ((-1)\cdot 8) \bmod 6$.

Beweis von Satz 1.1 für „+“:

Es gibt k, l mit $a_1 = a_2 + k \cdot m$, $b_1 = b_2 + l \cdot m$.

$\Rightarrow a_1 + b_1 = a_2 + k \cdot m + b_2 + l \cdot m = a_2 + b_2 + (k+l) \cdot m.$

Definition:

Zu $m > 0$ sei $\mathbb{Z}_m := \{0, 1, \ldots, m-1\}$.

Auf $\mathbb{Z}_m$ werden zwei Verknüpfungen $\oplus$ und $\odot$ definiert durch

$$a \oplus b := (a+b) \bmod m \qquad a \odot b := (a \cdot b) \bmod m.$$

Bemerkungen:

1. Alternative Interpretation von $\mathbb{Z}_m$:

   Die Menge aller Zahlen mit gleichem Rest bei Division durch m bezeichnet man als Restklasse:

   $$[a_0]_m := \{a \in \mathbb{Z} : a \equiv a_0 \bmod m\},$$

   z.B.

   $$[2]_6 = \{\ldots, -10, -4, 2, 8, 14, \ldots\}.$$

   Satz 1.1 besagt, dass für $a_1, a_2 \in [a_0]$, $b_1, b_2 \in [b_0]$ gilt:

   $$[a_1 + b_1] = [a_2 + b_2] \text{ und } [a_1 \cdot b_1] = [a_2 \cdot b_2].$$

   Ein $a \in [a_0]$ heißt Repräsentant.

   $\mathbb{Z}_m$ ist die Menge aller Restklassen, $\oplus$ und $\odot$ sind repräsentantenweise definiert. Statt $\mathbb{Z}_m$ schreibt man auch $\mathbb{Z}/m\mathbb{Z}$.

2. Man kann die Operationen in Verknüpfungstabellen darstellen, z.B. für $m = 4$:

$\oplus$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

$\odot$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

3. Bei Rechnungen in $\mathbb{Z}_m$ benötigt man keine großen Zahlen sondern kann stets reduzieren. Manchmal sind auch negative Zahlen hilfreich:
   Beispiel: Berechnung von $8 \odot 4 \odot (9 \oplus 5)$ in $\mathbb{Z}_{11}$:

$$\begin{aligned} 8 \odot 4 \odot (9 \oplus 5) &= [8 \cdot 4 \cdot (9+5)] \bmod 11 \\ &= [((8 \cdot 4) \bmod 11) \cdot ((9+5) \bmod 11)] \bmod 11 \\ &= [(32 \bmod 11) \cdot (14 \bmod 11)] \bmod 11 \\ &= [-1 \cdot 3] \bmod 11 \\ &= -3 \bmod 11 \;=\; 8. \end{aligned}$$

Definition:

Eine Menge G mit einer Verknüpfung $\circ : G \times G \to G$ heißt *Gruppe* genau dann, wenn gilt

1. $\circ$ ist assoziativ:

$$\forall\, a, b, c \in G : (a \circ b) \circ c = a \circ (b \circ c).$$

2. Es gibt ein neutrales Element $n \in G$:

$$\exists\, n \in G : \forall\, a \in G : a \circ n = n \circ a = a.$$

3. Zu jedem Element gibt es ein Inverses:

$$\forall\, a \in G\, \exists\, \bar{a} \in G : a \circ \bar{a} = \bar{a} \circ a = n.$$

G heißt *kommutative Gruppe* genau dann, wenn $\circ$ zusätzlich kommutativ ist, d.h

$$\forall a, b \in G : a \circ b = b \circ a.$$

Beispiele:

1. Folgendes sind kommutative Gruppen:
   - $\mathbb{Z}$ mit „+“,
   - $\mathbb{R}$ mit „+“,

- $\mathbb{R} \setminus \{0\}$ mit „$\cdot$“,
- $\mathbb{Z}_m$ mit „$\oplus$“.

2. $\mathbb{Z} \setminus \{0\}$ mit „$\cdot$“ ist keine Gruppe.
3. $\mathbb{Z}_4 \setminus \{0\}$ mit „$\odot$“ ist keine Gruppe.
4. $\mathbb{Z}_5 \setminus \{0\}$ mit „$\odot$“ ist eine kommutative Gruppe:

$\odot$	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

5. Die Menge aller invertierbaren 2×2-Matrizen bildet bzgl. „$\cdot$“ eine nichtkommutative Gruppe.

Satz 1.2:

Das neutrale Element sowie die inversen Elemente in einer Gruppe sind eindeutig.

Beweis für das neutrale Element:

Seien n_1 und n_2 neutrale Elemente. Zu zeigen: $n_1 = n_2$.

Da n_1 neutrales Element ist, gilt: $n_1 \circ n_2 = n_2$.
Da n_2 neutrales Element ist, gilt: $n_1 \circ n_2 = n_1$.

Also: $n_1 = n_1 \circ n_2 = n_2$.

Bemerkung:

In einer Gruppe sind Gleichungen der Form $c \circ x = d$ immer lösbar (c, d gegeben, x gesucht). Die Lösung erhält man durch Verknüpfung mit dem inversen Element:

$$c \circ x = d \quad \Rightarrow \quad \bar{c} \circ (c \circ x) = \bar{c} \circ d.$$

Wegen

$$\bar{c} \circ (c \circ x) = (\bar{c} \circ c) \circ x = n \circ x = x$$

folgt $x = \bar{c} \circ d$.

Die Betrachtung von $c \circ x$ für verschiedene x bedeutet in der Verknüpfungstabelle die Betrachtung verschiedener Einträge einer Zeile. Die eindeutige Lösbarkeit von $c \circ x = d$ bedeutet, dass in jeder Zeile jedes Element genau einmal vorkommt. Entsprechendes gilt auch für Spalten.

Bei den üblichen Rechenoperationen entspricht die obige Rechnung:

1. in $\mathbb{Z}$ oder $\mathbb{R}$ mit „+“:
   Das inverse Element zu c ist $-c$ und die Auflösung von $c+x=d$ geschieht durch Addition von $-c$ („Subtrahieren ist Addieren mit dem Negativen“),
2. in $\mathbb{R}\setminus\{0\}$ mit „$\cdot$“:
   Das inverse Element zu c ist $\frac{1}{c}$ und die Auflösung von $c\cdot x=d$ geschieht durch Multiplikation mit $\frac{1}{c}$ („Teilen durch c ist Multiplizieren mit dem Inversen zu c, also mit $\frac{1}{c}$“).

Satz 1.3:

Ist $(G,\circ)$ eine Gruppe mit endlich vielen Elementen (die Anzahl sei $|G|$), so gilt für jedes $x\in G$:

$$\underbrace{x\circ\ldots\circ x}_{|G|\text{-}mal}=n.$$

Beweis von Satz 1.3 für kommutatives G:

Sei $|G|=g$ und $G=\{a_1,\ldots,a_g\}$. Sei $x\in G$ beliebig. Dann sind $x\circ a_1,\ldots,x\circ a_g$ paarweise verschieden, denn aus $x\circ a_i=x\circ a_j$ würde durch Verknüpfung mit $\bar{x}$ folgen, dass $a_i=a_j$. Die Elemente $x\circ a_1,\ldots,x\circ a_g$ stellen also alle Elemente der Gruppe dar. Die Verknüpfung aller Elemente der Gruppe miteinander kann man also einerseits schreiben als $a_1\circ\ldots\circ a_g$, andererseits als $(x\circ a_1)\circ\ldots\circ(x\circ a_g)$. Damit gilt:

$$\begin{aligned}
& a_1\circ\ldots\circ a_g=(x\circ a_1)\circ\ldots\circ(x\circ a_g)\\
\Rightarrow\ & a_1\circ\ldots\circ a_g=a_1\circ\ldots\circ a_g\circ\underbrace{x\circ\ldots\circ x}_{g\text{-mal}}\\
\Rightarrow\ & \overline{(a_1\circ\ldots\circ a_g)}\circ a_1\circ\ldots\circ a_g=\overline{(a_1\circ\ldots\circ a_g)}\circ a_1\circ\ldots\circ a_g\circ\underbrace{x\circ\ldots\circ x}_{g\text{-mal}}\\
\Rightarrow\ & n=\underbrace{x\circ\ldots\circ x}_{g\text{-mal}}.
\end{aligned}$$

Bemerkung:

Der allgemeine Beweis von Satz 1.3 (auch für nicht-kommutative Gruppen) benutzt ganz andere Hilfsmittel.

Bemerkung:

Was besagt Satz 1.3 in der Gruppe $(\mathbb{Z}_m,\oplus)$?

Es ist $|\mathbb{Z}_m|=m$. Für jedes $x\in\mathbb{Z}_m$ gilt also

$$\underbrace{x\circ\ldots\circ x}_{m\text{-mal}}=m\cdot x \bmod m=0.$$

2 Public-Key Verfahren

Ein Nachteil symmetrischer Verschlüsselungsverfahren ist, dass mit jedem Gesprächspartner ein individueller Schlüssel vereinbart werden muss, der geheim übermittelt und geheim gespeichert werden muss.

Eine Alternative bietet das Public-Key Prinzip:

Vorbild - die gewöhnliche Briefpost:

> Jeder, der Post empfangen möchte, hat einen öffentlichen Briefkasten, den jeder kennt, und in den jeder (ohne vorher mit dem Besitzer gesprochen zu haben und ohne selbst einen Briefkasten haben zu müssen) einen Brief einwerfen kann. Nur der Besitzer des Briefkastens hat den Schlüssel zum Öffnen des Briefkastens.

Mathematisierung:

> Jeder, der geheime Post empfangen will, veröffentlicht eine Funktion, die leicht zu berechnen ist, deren Umkehrung aber schwierig ist. Der „Besitzer“ der Funktion hat allerdings ein Zusatzwissen, das eine effiziente Umkehrung der Funktion erlaubt.
>
> Genauer:
>
> Sei M die Menge der Klartexte / Nachrichten und C die Menge der Geheimtexte / codierten Nachrichten. Dann ist f eine injektive Funktion von M nach C, $f : M \to C$, die leicht berechenbar ist, so dass jeder einen Klartext $m \in M$ zu $c := f(m)$ verschlüsseln kann.
>
> Es ist schwierig, die Funktion umzukehren, d.h. zu einem $c \in C$ ein $m \in M$ zu berechnen mit $f(m) = c$. Mit Zusatzwissen ist diese Berechnung allerdings einfach.

Für einen allgemeinen Gebrauch ist es vorteilhaft, eine ganze Klasse solcher Funktionen zu haben. Für jedes $k \in K$ gibt es eine derartige Funktion f_k. Jedem, der geheime Post empfangen will, ist ein $k \in K$ zugeordnet. K und die Zuordnung $k \leftrightarrow$ Nutzer ist öffentlich (*Public-Key*). Mit einem individuellen zu k gehörigem Zusatzwissen l lässt sich f_k umkehren. l muss geheim gehalten werden (*Private-Key*).

3 Schwierige Probleme

Wie lange braucht ein Rechner zum Lösen verschiedener Probleme in $\mathbb{Z}_m$ in Abhängigkeit von der Größe von m?

Sei s die binäre Länge von m ($s \approx \operatorname{ld} m$, also $m \approx 2^s$).

Addition zweier Zahlen der Länge s:

```
      *   *   *   *   *
      *   *   *   *   *
  ---------------------
  *   *   *   *   *   *
```

Ca. $2s$ binäre Operationen.

Multiplikation zweier Zahlen der Länge s:

```
  *   *   *   *     ·       *   *   *   *
  -------------------------------------------
                        *   *   *   *
                    *   *   *   *
                *   *   *   *
            *   *   *   *
  -------------------------------------------
          *   *   *   *   *   *   *   *
```

Ca. s Additionen = ca. $2s^2$ binäre Operationen.

Reduktion einer $2s$-stelligen Zahl modulo m mit s-stelligem m:

```
  *   *   *   *   *   *     :     *   *   *
  *   *   *
  -------------------------
      *   *   *   *   *
      *   *   *
  -----------------
          *   *   *   *
          *   *   *
  -----------------
              *   *   *
```

Ca. s Subtraktionen = ca. $2s^2$ binäre Operationen.

Berechnung von $n^a \bmod m$ durch

$$n^a = (\ldots((n \cdot n \bmod m) \cdot n \bmod m)\ldots) \cdot n \bmod m$$

bei s-stelligen n, a und m:
Ca. a Multiplikationen und Reduktionen = ca. $a \cdot (2s^2 + 2s^2) \approx 2^s \cdot (2s^2 + 2s^2)$ binäre Operationen.

Beispiel:

Eine binäre Operation dauere eine Nanosekunde, d.h. 10^9 Operationen pro Sekunde.

	Dauer für		
Stellenzahl	s Operationen	s^2 Operationen	2^s Operationen
s=10	10 ns	0,1 μs	1 μs
s=100	0,1 μs	10 μs	> Alter des Universums
s=1000	1 μs	1 ms	

Definition:

Man sagt, ein Algorithmus hat eine Laufzeit $O(s)$ (lies „Groß-O von s“) bzw. $O(s^2)$, ..., wenn er in Abhängigkeit von der Problemgröße s eine Laufzeit maximal in der Größenordnung s bzw. s^2, ... besitzt.

Ein Problem nennt man *schwierig*, wenn es nicht in polynomialer Zeit (in Abhängigkeit von der Problemgröße) gelöst werden kann, d.h. es gibt kein r, so dass die Laufzeit $O(s^r)$ ist.

In der Zahlentheorie/Kryptologie wird die Länge einer Zahl (nicht die Größe der Zahl) als Problemgröße genommen.

Ist die Berechnung von $n^a \bmod m$ schwierig? Nein!

Idee:

$$\begin{aligned} 5^{16} \bmod 11 &= (((5^2)^2)^2)^2 \bmod 11 \\ &= (((25)^2)^2)^2 \bmod 11 = ((3^2)^2)^2 \bmod 11 \\ &= (9^2)^2 \bmod 11 \\ &= (81)^2 \bmod 11 = 4^2 \bmod 11 \\ &= 16 \bmod 11 = 5 \end{aligned}$$

(4 Multiplikationen und Reduktionen)

$$5^{18} \bmod 11 = 5^{16+2} \bmod 11 = 5^{16} \cdot 5^2 \bmod 11.$$

(4 Multiplikationen und Reduktionen plus eine Multiplikation und Reduktion)

$$5^{21} \bmod 11 = 5^{16+4+1} \bmod 11$$

Schnelle Exponentiation (Square-and-Multiply):

Zur Berechnung von $n^a \bmod m$ berechnet man

$$n^2 \bmod m, \quad n^4 = (n^2)^2 \bmod m, \quad n^8 = (n^4)^2 \bmod m, \ldots$$

bis n^{2^s}, wobei s die Stellenanzahl von a ist.

Entsprechend der Binärdarstellung von a multipliziert man die n^{2^t}.

Bei s-stelligem n, a und m benötigt man

maximal s Mult. und Red. (Berechnung von n^2, n^4, n^8, ..., n^{2^s})
\+ maximal s Mult. und Red. (Mult. der n^{2^t} entspr. der binären Darstellung von a)
= maximal $2s$ Multiplikationen und Reduktionen
≈ $2s \cdot (2s^2 + 2s^2) = 8s^3$ binäre Operationen

Schwierige Probleme können als Grundlage kryptografischer Algorithmen dienen.

Frage:

Welche der folgenden Probleme sind schwierig?

(1) Auflösen von $cx = d \bmod m$ (c, d, m gegeben) nach x,
z.B. $3x = 4 \bmod 7$.

(2) Finden eines x mit $a^x = d \bmod m$ (a, d, m gegeben),
z.B. $3^x = 4 \bmod 7$.

(3) Auflösen von $x^2 = a \bmod m$ (a, m gegeben) nach x,
z.B. $x^2 = 2 \bmod 7$.

(4) Entscheidung, ob ein gegebenes m eine Primzahl ist,

(5) Auffinden eines echten Teilers einer gegebenen Zahl m.

Antwort:

Keines dieser Probleme ist als schwierig nachgewiesen (Stand Sommer 2022). Für (1) und (4) gibt es effiziente Algorithmen, bei (3) gibt es sie, falls m eine Primzahl ist oder das Produkt zweier Primzahlen, die man kennt. Diese Probleme sind nicht schwierig. Für die anderen Probleme kennt man bisher keine effizienten Algorithmen. Ob sie tatsächlich schwierig sind, ist eine der großen offenen Fragen in der Informatik.

4 Der Euklidische Algorithmus

Ziel: Lösen von $cx = d \bmod m$, d.h. finde x, y mit $cx + my = d$.

Beispiele:

1. $8x = 7 \bmod 14$ bzw. $8x + 14y = 7$
   hat keine Lösung, da die linke Seite durch 2 teilbar ist, die rechte aber nicht.
2. $12x = 8 \bmod 18$ bzw. $12x + 18y = 8$
   hat keine Lösung, da die linke Seite durch 6 teilbar ist, die rechte aber nicht.

Eine notwendige Bedingung zur Lösbarkeit von $cx + my = d$ ist, dass der größte gemeinsame Teiler von c und m auch d teilt.

Erstes Teilziel: Bestimmung des größten gemeinsamen Teilers zweier Zahlen:

Definition:

Mit $\gcd(a, b)$ („greatest common divisor“) wird der größte gemeinsame Teiler von a und b bezeichnet.

Methode 1 - Bestimmung über gemeinsame Primfaktoren:

Beispiel:

Bestimmung von gcd(84,30):

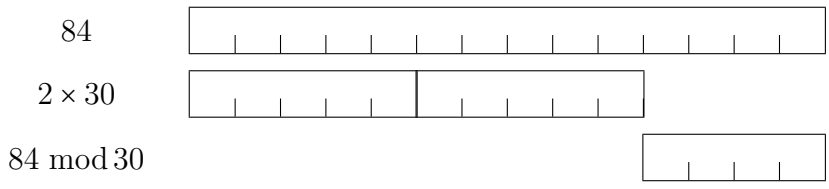
84	=	2	·	2	·	3	·	7	
30	=			2	·	3	·	5	
$\gcd(84, 30)$	=			2	·	3			$= 6$

Aber: Primfaktorzerlegung ist ein (wahrscheinlich) schwieriges Problem.

Methode 2 - Euklidischer Algorithmus:

Idee:

Idee: $\gcd(84, 30)$ teilt auch $84 \bmod 30$:



Satz 4.1

Es gilt: $\gcd(a, b) = \gcd(b, a \bmod b)$.

Der *euklidische Algorithmus* bezeichnet die iterative Berechnung des gcd mittels Satz 4.1:

$$\begin{aligned} r_0 &= a \\ r_1 &= b \\ r_{k+1} &= r_{k-1} \bmod r_k. \end{aligned}$$

Beispiel:

Berechnung von $\gcd(84, 30)$:

r_0	84
r_1	30
r_2	$84 \bmod 30 = 24$
r_3	$30 \bmod 24 = 6$
r_4	$24 \bmod 6 = 0$

Erreicht man $r_{k+1} = 0$, so gibt r_k den gcd an.

Zweites Teilziel: Lösen von $ax + by = \gcd(a, b)$:

Erweiterter Euklidischer Algorithmus:

Initialisierung:

$$r_0 = a, \qquad x_0 = 1, \qquad y_0 = 0,$$

$$r_1 = b, \qquad x_1 = 0, \qquad y_1 = 1,$$

Rekursion:

$$\begin{aligned} r_{k+1} &= r_{k-1} \bmod r_k \\ q_{k+1} &= r_{k-1} \operatorname{div} r_k \\ x_{k+1} &= x_{k-1} - x_k \cdot q_{k+1} \\ y_{k+1} &= y_{k-1} - y_k \cdot q_{k+1} \end{aligned}$$

(Dabei bezeichnet $a \operatorname{div} b$ den ganzzahligen Teil bei der Division von a durch b, z.B. $84 \operatorname{div} 30 = 2$.)

Durch Induktion lässt sich zeigen:

$$r_k = x_k \cdot a + y_k \cdot b.$$

Erreicht man die Zeile, in der r_k den gcd angibt (also $r_{k+1} = 0$), so erhält man mit $x = x_k$ und $y = y_k$ eine Lösung zu $ax + by = \gcd(a, b)$.

Beispiel:

Lösen von $84x + 30y = \gcd(84, 30) = 6$

k	r_k	q_k	x_k	y_k
0	84		1	0
1	30		0	1
2	24	2	1	-2
3	6	1	-1	3

Für $x = -1$ und $y = 3$ gilt

$$84x + 30y \;=\; -84 + 30 \cdot 3 \;=\; 6.$$

Ferner gilt beispielsweise ($k = 2$): $r_2 = 24 = 1 \cdot 84 + (-2) \cdot 30$.

Algorithmus zum Lösen von $cx \equiv d \bmod m$:

1. Berechne $g := \gcd(c, m)$.
2. Falls $g \nmid d$: Keine Lösung. - Ende
3. Finde $(\hat{x}, \hat{y})$ mit $c\hat{x} + m\hat{y} = g$.
4. $\tilde{x} = \frac{d}{g}\hat{x}$ ist dann eine Lösung von $c\tilde{x} \equiv d \bmod m$.

   Wegen 3. gilt nämlich

   $$d = \frac{d}{g} \cdot g = \frac{d}{g} \cdot c \cdot \hat{x} + \frac{d}{g} \cdot m \cdot \hat{y} = c \cdot \tilde{x} + \left(\frac{d}{g} \cdot \hat{y}\right) \cdot m.$$

5. $x = \tilde{x} \bmod m$ ist dann eine Lösung von $cx \equiv d \bmod m$, die in $\{0, \ldots, m-1\}$ liegt.

Beispiel:

Löse $12x = 52 \bmod 56$:

1. Euklidischer Algorithmus:

r_0	56
r_1	12
r_2	$56 \bmod 12 = 8$
r_3	$12 \bmod 8 = 4$
r_4	$8 \bmod 4 = 0$

   $\Rightarrow \gcd(12, 56) = 4$.

2. $4 \mid 52$, d.h. die Aufgabe hat eine Lösung.

3. Erweiterter euklidischer Algorithmus zu $12x + 56m = 4$

k	r_k	q_k	x_k	y_k
0	12		1	0
1	56		0	1
2	12	0	1	0
3	8	4	-4	1
4	4	1	5	-1

$\Rightarrow\ 12 \cdot 5 + 56 \cdot (-1) = 4$, also $\hat{x} = 5$ (und $\hat{y} = -1$).

4. $\tilde{x} = \frac{52}{4} \cdot 5 = 13 \cdot 5 = 65$.

   Tatsächlich ist $12 \cdot 65 \equiv 52 \bmod 56$.

5. Damit erfüllt auch $x = 65 \bmod 56 = 9$ die Gleichung:

$$12 \cdot 9 = 108 \equiv 52 \bmod 56.$$

Bemerkung:

Da bei 4. nur $\hat{x}$ und nicht $\hat{y}$ genutzt wird, braucht man $\hat{y}$ in 3. auch gar nicht zu berechnen. Die entsprechende Spalte beim erweiterten Euklidischen Algorithmus kann man also sparen.

Beginnt man wie im Beispiel mit der kleineren Zahl beim erweiterten euklidischen Algorithmus, ändert sich zunächst kaum etwas, außer dass sich die Rollen vertauschen.

Merkregel:

Nutze beim erweiterten euklidischen Algorithmus eine Spalte, die so mit 1 und 0 initialisiert ist, dass die 1 bei der Zahl steht, die der Koeffizient bei der zu lösenden Gleichung ist (im Beispiel die 12).

Ist $g = \gcd(c, m) = 1$, so gilt für $\hat{x}$ in Schritt 3

$$c \cdot \hat{x} + m\hat{y} = 1 \quad \Rightarrow \quad c \cdot \hat{x} = 1 \bmod m \quad \Rightarrow \quad \hat{x} = c^{-1} \bmod m.$$

Eine Lösung zu $cx = d \bmod m$ ergibt sich dann entspr. 4. direkt durch

$$x \ = \ c^{-1} \cdot d \ = \ \hat{x} \cdot d \quad (\bmod m).$$

Zusammenfassung der Lösbarkeit:

Satz 4.2:

1. *Die Gleichung $c \cdot x = d \bmod m$ ist lösbar* $\Leftrightarrow$ $\gcd(c, m)$ *teilt d.*
2. *Die Gleichung $c \cdot x = 1 \bmod m$ ist lösbar* $\Leftrightarrow$ $\gcd(c, m) = 1$.
   (Dann ist $x = c^{-1} \bmod m$ und wird in Schritt 3 des erw. eukl. Alg. berechnet.)

Laufzeitbetrachtung beim Euklidischen Algorithmus:

Satz:

Beim Euklidischen Algorithmus hat sich a nach 2 Schritten mindestens halbiert.

Beweis:

Zu zeigen: Für $b < a$ ist $a \bmod b \leq \frac{a}{2}$. $(*)$

1. Fall: $b \leq \frac{a}{2}$: Wegen $a \bmod b \leq b$ folgt $(*)$.

2. Fall: $b > \frac{a}{2}$: Dann gilt

$$a \bmod b \leq a - b < a - \frac{a}{2} = \frac{a}{2} \quad \Rightarrow \quad (*).$$

Bei $a, b \approx 2^s$ benötigt man also höchstens $2s$ Schritte, bei $a, b \approx 10^s = 2^{(\mathrm{ld}\,10)s}$ also höchstens $2 \cdot (\mathrm{ld}\,10)s \approx 6.7s$ Schritte.

Bemerkung:

Man kann zeigen, dass bei a, b gleich zwei aufeinanderfolgende Fibonacci-Zahlen $(1, 1, 2, 3, 5, 8, 13, \ldots)$ maximale Schrittanzahl auftritt (bei $a, b \approx 10^s$ ca. $4.8s$ Schritte).

5 Ringe und Körper

$\mathbb{Z}_m$ ist bzgl. $\oplus$ eine kommutative Gruppe. In $\mathbb{Z}_m$ gibt es noch die zweite Verknüpfung $\odot$, die sich mit $\oplus$ verträgt.

Definition:

Eine Menge R mit zwei Verknüpfungen $\boxplus$ und $\boxdot$ heißt *Ring* genau dann, wenn gilt

1. $(R, \boxplus)$ ist eine kommutative Gruppe.
2. $\boxdot$ ist assoziativ.
3. Es gilt das Distributivgesetz:

$$\begin{aligned} \forall\, a,b,c \in R: \quad & (a \boxplus b) \boxdot c = (a \boxdot c) \boxplus (b \boxdot c) \\ \text{und} \quad & a \boxdot (b \boxplus c) = (a \boxdot b) \boxplus (a \boxdot c). \end{aligned}$$

Ist $\boxdot$ zusätzlich kommutativ, so heißt R *kommutativer Ring.*
Gibt es ein Element $e \in R$, so dass gilt

$$\forall\, a \in R : a \boxdot e = e \boxdot a = a,$$

so heißt R *Ring mit Einselement.*

Bemerkung:

Das neutrale Element n bzgl. $\boxplus$ heißt auch *Nullelement.*

Beispiele:

1. $\mathbb{Q}$, $\mathbb{R}$, $\mathbb{C}$, $\mathbb{Z}$, $\mathbb{Z}_m$ sind kommutative Ringe mit Einselement.
2. Die Menge aller 2×2-Matrizen ist mit der üblichen Matrizenaddition und -multiplikation ein (nicht kommutativer) Ring mit Einselement, $e = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$.
3. Die Menge aller geraden Zahlen ist mit der üblichen Addition und Multiplikation ein kommutativer Ring (ohne Einselement).

Gelten Gesetze, wie sie in den reellen Zahlen üblich sind, in jedem Ring, z.B. $x \cdot 0 = 0$ oder $x \cdot y = 0 \Rightarrow x = 0 \vee y = 0$? Letzeres sicherlich nicht, denn z.B. gilt im Ring $\mathbb{Z}_6$: $2 \odot 3 = 0$.

Satz 5.1:

In einem Ring R gilt für alle $x \in R$: $x \boxdot n = n \boxdot x = n$.

Beweis:

$x \boxdot n = x \boxdot (n \boxplus n) = (x \boxdot n) \boxplus (x \boxdot n) \;\Rightarrow\; n = x \boxdot n.$

In einem kommutativen Ring R mit Einselement ist $\boxdot$ kommutativ und assoziativ und e ist ein neutrales Element bzgl. $\boxdot$ für alle Elemente $x \neq n$. Damit ist R bzgl. $\boxdot$ schon fast eine Gruppe. Es fehlen nur noch die inversen Elemente. Wegen Satz 5.1 kann es allerdings zu n kein inverses Element geben.

Definition:

Eine Menge K mit zwei Verknüpfungen $\boxplus$ und $\boxdot$ heißt *Körper* genau dann, wenn gilt

1. $(K, \boxplus, \boxdot)$ ist ein kommutativer Ring mit Einselement e.
2. Für alle $a \in K$, $a \neq n$ gibt es ein Inverses $a^{-1} \in K$ bzgl. $\boxdot$:

$$\forall\, a \in K, a \neq n : \exists\, a^{-1} \in K : a \boxdot a^{-1} = e.$$

Beispiele:

1. $\mathbb{Q}$, $\mathbb{R}$ und $\mathbb{C}$ sind Körper.
2. $\mathbb{Z}$ ist kein Körper.
3. Wann ist $\mathbb{Z}_m$ ein Körper?

$$\begin{aligned}
& \mathbb{Z}_m \text{ ist ein Körper} \\
\Leftrightarrow\quad & \text{für jedes } c \in \mathbb{Z}_m,\ c \neq 0 \text{ gibt es ein } x \text{ mit } c \odot x = 1 \\
\Leftrightarrow\quad & \text{für jedes } c \in \{1, \ldots, m-1\} \text{ ist } c \cdot x = 1 \bmod m \text{ lösbar} \\
\overset{\text{Satz4.2}}{\Leftrightarrow}\quad & \text{für jedes } c \in \{1, \ldots, m-1\} \text{ ist } \gcd(c, m) = 1 \\
\Leftrightarrow\quad & m \text{ ist eine Primzahl.}
\end{aligned}$$

Satz 5.2:

$\mathbb{Z}_m$ ist ein Körper $\Leftrightarrow$ m ist eine Primzahl.

Vereinbarung:

In Anlehnung an $\mathbb{R}$ schreibt man bei einem beliebigen Ring oder Körper statt „$\boxplus$“ auch einfach „$+$“ (Addition), statt „$\boxdot$“ einfach „$\cdot$“ (Multiplikation), für die neutralen Elemente 0 bzw. 1 statt n bzw. e, für das additive Inverse „$-x$“, für das multiplikative Inverse „x^{-1}“ und z.B. „$a-b$“ statt „$a+(-b)$“. Ferner gelte Punkt- vor Strichrechnung.

Ebenso schreibt man bei Rechnungen in $\mathbb{Z}_m$ nur „$+$“ bzw. „$\cdot$“ statt „$\oplus$“ bzw. „$\odot$“.

Satz 5.3:

In einem Körper K gilt für alle $x, y \in K$:

$$x \cdot y = 0 \;\Rightarrow\; x = 0 \vee y = 0.$$

(Man sagt: Ein Körper ist nullteilerfrei.)

Beweis:

Sei $x \cdot y = 0$ und $x \neq 0$. Zu zeigen ist $y = 0$.

Wegen $x \neq 0$ existiert x^{-1} und es folgt

$$y = 1 \cdot y = x^{-1} \cdot x \cdot y = x^{-1} \cdot 0 \overset{\text{Satz5.1}}{=} 0.$$

Ist $(K, +, \cdot)$ ein Körper, so ist $(K \setminus \{0\}, \cdot)$ eine Gruppe. Satz 5.3 gewährleistet, dass die Multiplikation nicht aus $K \setminus \{0\}$ herausführt.

6 Primzahltest

Problem:

Entscheide, ob eine gegebene *ungerade* Zahl n eine Primzahl ist.

Lösungsideen:

1. Probedivision:

   $$n \text{ ist eine Primzahl} \Leftrightarrow \forall k, k \leq \sqrt{n} \text{ gilt } k \nmid n.$$

   Aufwand: $\sqrt{n}$ viele Berechnungen.

   Bei $n \approx 2^s$ ist $\sqrt{n} \approx 2^{s/2} = (\sqrt{2})^s$, d.h., der Aufwand ist exponentiell.

2. Der kleine Satz von Fermat (Satz 7.2) besagt:

   $$p \text{ ist eine Primzahl} \Rightarrow \forall a \text{ mit } \gcd(a,p) = 1 \text{ gilt } a^{p-1} = 1 \bmod p. \quad (*)$$

   Fermattest zur Überprüfung, ob n eine Primzahl ist:
   Ziehe ein $a \in \{1, \ldots, n-1\}$ und prüfe, ob $\gcd(a,n) = 1$ und $a^{n-1} = 1 \bmod n$ ist. Falls nicht, ist n keine Primzahl.

   Die Umkehrung von $(*)$ gilt allerdings nicht. Es gibt Zahlen, für die die rechte Seite von $(*)$ gilt, obwohl sie keine Primzahlen sind. Solche Zahlen heißen *Carmichael Zahlen.* Die kleinsten Carmichael Zahlen sind 561, 1105, 1729. Es gibt unendlich viele Carmichael Zahlen.

   Mit dem Fermattest kann man also nur zeigen, dass n *keine* Primzahl ist.

Probabilistischer Test:

Ziel: Finde eine Eigenschaft $E(a)$ mit

$$n \text{ ist eine Primzahl} \Rightarrow \forall a = 1, \ldots, n-1 \text{ gilt } E(a)$$

und

$$\begin{array}{rl} & n \text{ ist keine Primzahl} \\ \Rightarrow & \text{für mindestens die Hälfte aller } a \in \{1, \ldots, n-1\} \text{ gilt } E(a) \text{ nicht.} \end{array}$$

Dann kann man einen Test wie folgt durchführen:

1. Setze eine Irrtumswahrscheinlichkeit ε oder Maximalzahl K fest, $k := 1$.
2. Ziehe zufällig $a \in \{1, \ldots, n-1\}$.
3. Falls $E(a)$ nicht gilt:
   Ausgabe: n ist sicher keine Primzahl.
   Ende.

4. Falls $\left(\frac{1}{2}\right)^k < \varepsilon$ oder $k = K$:
   Ausgabe: n ist wahrscheinlich eine Primzahl.
   Ende.

5. Erhöhe k um 1, gehe zu 2..

Für $\varepsilon = 10^{-100}$ sind ca. 333 Iterationen nötig ($\left(\frac{1}{2}\right)^{333} < 10^{-100}$).

Eine geeignete Eigenschaft $E(a)$:

Ist n eine Primzahl, so ist $\mathbb{Z}_n$ ein Körper und die Gleichung $x^2 = 1 \bmod n$ hat nur die Lösungen 1 und -1 ($\bmod n$) (s. Übung).

Damit kann man den Fermattest erweitern: Ist $a^{n-1} \neq 1 \bmod n$, so ist n keine Primzahl. Ist hingegen $a^{n-1} = 1 \bmod n$, so kann man $x = a^{\frac{n-1}{2}}$ betrachten. Es ist

$$x^2 = \left(a^{\frac{n-1}{2}}\right)^2 = a^{n-1} = 1 \bmod n.$$

Wenn n eine Primzahl ist, muss also $x = 1$ oder $x = -1$ sein. Ist $x \neq \pm 1 \bmod n$, so ist n sicher keine Primzahl.

Ist $x = 1$ und $\frac{n-1}{2}$ gerade, kann man weiter $a^{\frac{n-1}{4}}$ betrachten und so fort.

Damit kommt man zu folgendem Algorithmus zu einem gegebenen $a \in \{1, \ldots, n-1\}$:

1. Falls $a^{n-1} \neq 1 \bmod n$:
   Ausgabe: a ist ein Zeuge für die Zusammengesetztheit von n.
   Ende.

2. Setze $k := \frac{n-1}{2}$.

3. Falls $a^k \neq \pm 1 \bmod n$:
   Ausgabe: a ist ein Zeuge für die Zusammengesetztheit von n.
   Ende.

4. Falls $a^k = -1 \bmod n$ oder k ungerade ist:
   Ausgabe: möglicherweise prim.
   Ende.

5. Setze $k := \frac{k}{2}$, gehe zu 3..

Die Eigenschaft $E(a)$ ist die Ausgabe „möglicherweise prim“ in dem Algorithmus. Wenn n eine Primzahl ist, erhält man für jedes a die Ausgabe „möglicherweise prim“, d.h. $E(a)$ ist erfüllt. Ist n hingegen zusammengesetzt, sind beide Ausgaben („Zeuge für die Zusammengesetztheit“ oder „möglicherweise prim“) möglich. Man kann allerdings zeigen, dass mindestens die Hälfte (sogar mindestens drei Viertel) aller $a \in \{1, \ldots, n-1\}$ die Ausgabe „Zeuge für die Zusammengesetztheit“ erzeugt, d.h. $E(a)$ gilt nicht. (Tatsächlich ist die Eigenschaft *nicht* $E(a)$ bei zusammengesetztem n meist noch deutlich häufiger.)

Den probabilistischen Primzahltest mit der hier beschriebenen Eigenschaft $E(a)$ nennt man *Miller-Rabin-Test*.

Beispiel:

Betrachte $n = 561$.

Zu $a = 50$ ergibt sich:

$$a^{560} = 1 \bmod 561, \quad a^{280} = 1 \bmod 561, \quad a^{140} = 1 \bmod 561,$$

$$a^{70} = 1 \bmod 561, \quad a^{35} = -1 \bmod 561.$$

Damit ist 50 kein Zeuge für die Zusammengesetztheit.

Zu $a = 2$ ergibt sich:

$$a^{560} = 1 \bmod 561, \quad a^{280} = 1 \bmod 561, \quad a^{140} = 67 \bmod 561.$$

Damit ist 2 ein Zeuge für die Zusammengesetztheit.

Alternative Möglichkeit zur Prüfung von $E(a)$:

Dem obigen Algorithmus liegt folgende Struktur zu Grunde: :

Ist n eine ungerade Primzahl, so gilt für jedes $a \in \{1, \ldots, n-1\}$

$$\text{mit } k := n-1: \quad a^k = 1 \bmod n$$

und damit

$$\text{mit } k := \frac{k}{2}: \quad a^k = 1 \bmod n \quad \text{oder} \quad a^k = -1 \bmod n.$$

Im ersten Fall (also $a^k = 1 \bmod n$) folgt weiter

$$\text{mit } k := \frac{k}{2}: \quad a^k = 1 \bmod n \quad \text{oder} \quad a^k = -1 \bmod n$$

u.s.w. bis k ungerade ist.

Ein „Ausstieg“ erfolgt bei der untersten Zeile oder bei einer -1-Kongruenz.

Betrachtet man die Struktur von unten nach oben, so kann man umgekehrt folgern, dass ein „Einstieg“ in der untersten Zeile oder bei einer -1-Kongruenz erfolgen muss. Damit ergibt sich eine alternative Möglichkeit der Prüfung:

Sei $n - 1 = 2^{d_0} \cdot r$ mit r ungerade (r entspricht dann dem letzten k der Struktur). Ist n eine Primzahl, so gilt:

$$a^r = 1 \bmod n \quad \text{oder} \quad \text{für ein } d \in \{0, \ldots, d_0 - 1\} \text{ ist } a^{2^d \cdot r} = -1 \bmod n \qquad (*)$$

Eine Zahl a ist genau dann Zeuge für die Zusammengesetztheit von n, wenn $(*)$ nicht gilt.

Die Berechnung von $a^{2^d \cdot r}$ für wachsendes d entspricht dabei sukzessivem Quadrieren. Sobald man eine 1 erhält, ohne vorher eine -1 gehabt zu haben, kann man die Berechnung mit der Ausgabe „Zeuge für die Zusammengesetztheit“ beenden, denn weiteres Quadrieren verbleibt dann bei 1.

Beispiel:

Betrachte $n = 561$. Es ist $n - 1 = 560 = 2^4 \cdot 35$

Zu $a = 50$ ergibt sich:

$$a^{35} = -1 \bmod 561.$$

Damit ist 50 kein Zeuge für die Zusammengesetztheit.

Zu $a = 2$ ergibt sich:

$$a^{35} = 263 \bmod 561, \quad a^{70} = 263^2 = 166 \bmod 561,$$

$$a^{140} = 166^2 = 67 \bmod 561, \quad a^{280} = 67^2 = 1 \bmod 561.$$

Damit ist 2 ein Zeuge für die Zusammengesetztheit.

Algorithmus zum Finden großer Primzahlen:

Wähle zufällig eine ungerade Zahl $n > 2$ und setze eine Irrtumswahrscheinlichkeit ε oder eine Maximalzahl K fest.

1. Führe den Miller-Rabin-Test für n mit der Irrtumswahrscheinlichkeit ε bzw. der Maximalzahl K durch.
2. Falls n (wahrscheinlich) prim ist: Ende.
3. $n := n + 2$, gehe zu 1..

Ist dieser Algorithmus endlich? Wenn ja: Wieviel Iterationen sind (im Durchschnitt) nötig?

Satz 6.1:

1. Es gibt unendlich viele Primzahlen.

2. Es gibt ungefähr $\frac{n}{\ln n}$ viele Primzahlen kleiner als n.

Die erste Aussage des Satzes besagt, dass der Algorithmus zum Finden großer Primzahlen endlich ist.

Die zweite Aussage besagt, dass durchschnittlich jede $(\ln n)$-te Zahl eine Primzahl ist. Die Anzahl der benötigten Iterationen wächst also ungefähr proportional zur Stellenanzahl.

Bemerkung:

Seit dem Jahr 2002 ist ein deterministischer Algorithmus mit polynomialer Laufzeit zur Bestimmung, ob eine gegebene Zahl n eine Primzahl ist, bekannt. Da dieser Algorithmus allerdings kompliziert ist, wird in der Praxis meist ein probabilistischer Algorithmus wie der Miller-Rabin-Test zur Bestimmung von Primzahlen benutzt.

7 Die Einheitengruppe

In einem kommutativen Ring mit Einselement kann man die Elemente sammeln, die ein multiplikatives Inverses besitzen. Die Menge dieser Elemente ist dann bzgl. „$\cdot$“ eine Gruppe:

Satz 7.1:

Ist $(R, +, \cdot)$ ein kommutativer Ring mit Einselement, so ist

$$R^\times := \{x \in R \mid \exists\, y \in R : x \cdot y = 1\}$$

bzgl. „$\cdot$“ eine kommutative Gruppe.

Man nennt $R^\times$ Einheitengruppe.

Beweis für die Abgeschlossenheit:

Zu zeigen ist, dass mit $x_1, x_2 \in R^\times$ auch $x_1 \cdot x_2 \in R^\times$ gilt.

Sind $x_1, x_2 \in R^\times$, so gibt es y_1 und y_2 mit

$$x_1 \cdot y_1 = 1 = x_2 \cdot y_2.$$

Damit gilt

$$(x_1 \cdot x_2) \cdot (y_1 \cdot y_2) = x_1 \cdot y_1 \cdot x_2 \cdot y_2 = 1 \cdot 1 = 1,$$

d.h. auch $x_1 \cdot x_2$ besitzt ein multiplikatives Inverses, ist also in $R^\times$ enthalten.

Beispiele:

1. Jeder Körper K ist ein kommutativer Ring mit Einselement.
   Es ist $K^\times = K \setminus \{0\}$.
2. $\mathbb{Z}^\times = \{-1, 1\}$.
3. Was ist ${\mathbb{Z}_m}^\times$?

   Es gilt:

   $$a \in {\mathbb{Z}_m}^\times \;\Leftrightarrow\; a \cdot y = 1 \bmod m \text{ ist lösbar} \overset{\text{Satz4.2}}{\Leftrightarrow} \gcd(a, m) = 1,$$

   also: ${\mathbb{Z}_m}^\times = \{a \in \mathbb{Z}_m \mid \gcd(a, m) = 1\}$.

m	Menge	Elemente	Anz.
2	$\mathbb{Z}_2{}^\times$	$\{1\}$	1
3	$\mathbb{Z}_3{}^\times$	$\{1,2\}$	2
4	$\mathbb{Z}_4{}^\times$	$\{1,3\}$	2
5	$\mathbb{Z}_5{}^\times$	$\{1,2,3,4\}$	4
6	$\mathbb{Z}_6{}^\times$	$\{1,5\}$	2
7	$\mathbb{Z}_7{}^\times$	$\{1,2,3,4,5,6\}$	6
8	$\mathbb{Z}_8{}^\times$	$\{1,3,5,7\}$	4
9	$\mathbb{Z}_9{}^\times$	$\{1,2,4,5,7,8\}$	6
10	$\mathbb{Z}_{10}{}^\times$	$\{1,3,7,9\}$	4
11	$\mathbb{Z}_{11}{}^\times$	$\{1,2,3,4,5,6,7,8,9,10\}$	10
12	$\mathbb{Z}_{12}{}^\times$	$\{1,5,7,11\}$	4

Erinnerung:

Satz 1.3 besagte:

Ist $(G,\circ)$ eine Gruppe mit endlich vielen Elementen (die Anzahl sei $|G|$), so gilt für jedes $x \in G$:

$$\underbrace{x \circ \ldots \circ x}_{|G|\text{-mal}} = n.$$

Beispiel:

$G = \mathbb{Z}_{10}{}^\times = \{1,3,7,9\}$, also $|G| = 4$. Es ist

$$\begin{aligned} 1^4 &= 1, \\ 3^4 &= 9^2 = 81 = 1 \bmod 10, \\ 7^4 &= 49^2 = 9^2 = 81 = 1 \bmod 10, \\ 9^4 &= 81^2 = 1^2 = 1 \bmod 10. \end{aligned}$$

Definition:

Zu $m > 1$ sei $\phi(m) := |\mathbb{Z}_m{}^\times|$.
ϕ heißt *Eulersche Phi-Funktion.*

Bemerkung:

Für eine Primzahl p ist $\phi(p) = |\mathbb{Z}_p{}^\times| = |\{1,\ldots,p-1\}| = p-1$.

Satz 7.2:

1. *Für jedes $m > 1$ und a mit $\gcd(a,m) = 1$ gilt $a^{\phi(m)} = 1 \bmod m$.*
   (Satz von Euler)
2. *Für jede Primzahl p und $a \in \{1,\ldots,p-1\}$ gilt $a^{p-1} = 1 \bmod p$.*
   (Kleiner Satz von Fermat)

Beispiel:

1. Zu $m = 10$ ist $\phi(m) = 4$. Es ist $3^4 = 81 = 1 \bmod 10$.

2. $p = 17$ ist eine Primzahl. Es gilt modulo 17:

$$6^{16} = 36^8 = 2^8 = 4^4 = 16^2 = (-1)^2 = 1 \bmod 17.$$

Satz 7.3:

Ist $m = {p_1}^{k_1} \cdot {p_2}^{k_2} \cdot \ldots \cdot {p_r}^{k_r}$ *mit verschiedenen Primzahlen* $p_1, \ldots, p_r$*, so ist*

$$\phi(m) = m \cdot \prod_{i=1}^{r} \left(\frac{p_i - 1}{p_i} \right).$$

Beispiele:

1. Berechnung von $\phi(12)$:
   Es ist $12 = 2^2 \cdot 3$, also $\phi(12) = 12 \cdot \frac{1}{2} \cdot \frac{2}{3} = 4$.
2. Berechnung von $\phi(8)$:
   Es ist $8 = 2^3$, also $\phi(8) = 8 \cdot \frac{1}{2} = 4$.

Folgerungen aus Satz 7.3:

1. Für das Produkt $n = p \cdot q$ zweier Primzahlen p und q ist

$$\phi(n) = p \cdot q \cdot \frac{p-1}{p} \cdot \frac{q-1}{q} = (p-1)(q-1).$$

2. Sind m_1 und m_2 teilerfremd, so gilt $\phi(m_1 \cdot m_2) = \phi(m_1) \cdot \phi(m_2)$.

8 RSA

RSA ist ein Public-Key - Verfahren, benannt nach den Entdeckern Ron Rivest, Adi Shamir und Len Adleman.

Schlüsselerzeugung:

Wähle zwei (große) verschiedene Primzahlen p und q.

Setze $n = p \cdot q$ und $f = (p-1) \cdot (q-1) = \phi(n)$.

Wähle ein e mit $\gcd(e, f) = 1$.

Berechne $d = e^{-1} \bmod f$.

Öffentlicher Schlüssel: n, e

Privater Schlüssel: d.

Ver- und Entschlüsselung:

Ein Klartext liege als Zahl $m \in \mathbb{Z}_n$ vor.

Verschlüsselung: $c = m^e \bmod n$.

Entschlüsselung: $m = c^d \bmod n$.

Warum ist $m = c^d \bmod n$?

Es ist

$$c^d = (m^e)^d = m^{ed} \quad \text{und} \quad ed = 1 \bmod f,$$

also

$$ed = k \cdot f + 1 = k \cdot \phi(n) + 1.$$

Für m teilerfremd zu n gilt nach dem Satz von Euler (Satz 7.2)

$$m^{\phi(n)} = 1 \bmod n,$$

also

$$m^{ed} = m^{k \cdot \phi(n)+1} = \left(m^{\phi(n)}\right)^k \cdot m = 1^k \cdot m = m \bmod n.$$

(Um $m = c^d \bmod n$ für nicht teilerfremde m und n zu zeigen, braucht man weitere Hilfsmittel.)

Bemerkungen:

1. Alle Operationen sind einfach durchführbar.
2. Die Sicherheit von RSA liegt an der vermutlichen Schwierigkeit

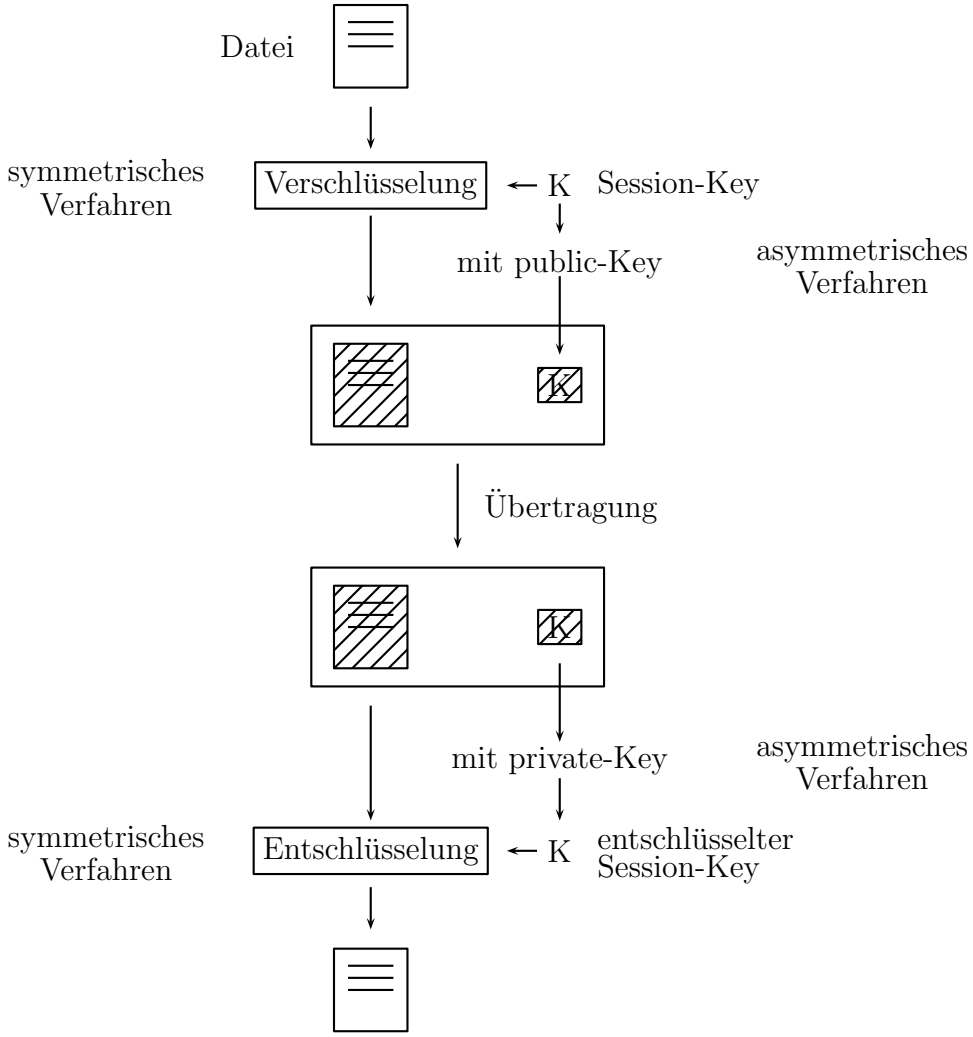
a) des Faktorisierungsproblems:
   Finde zu n echte Teiler.

b) der Invertierung der modularen Potenzfunktion:
   Finde zu gegebenen e, n und c ein x mit $x^e = c \bmod n$.

Zu diesen Problemen sind zur Zeit (Stand Herbst 2023) keine Algorithmen mit polynomialer Berechnungszeit bekannt. Es ist aber auch nicht bewiesen, dass es solche Algorithmen nicht gibt.

3. Der sogenannte *RSA-Modul* n liegt heutzutage ungefähr bei 2048 Bits, was einer ca. 600-stelligen Dezimalzahl entspricht. Deutlich weniger gilt heutzutage als unsicher.

4. Eine RSA-Verschlüsselung ist deutlich langsamer als eine Verschlüsselung mit symmetrischen Algorithmen. Man benutzt daher üblicherweise ein *Hybrid-Verfahren*: Der Sender wählt zufällig einen Schlüssel für ein symmetrisches Verfahren und verschlüsselt damit die zu übermittelnde Nachricht. Nur der Schlüssel wird dann mit dem Public-Key-Verfahren verschlüsselt.



Common-Modulus-Attacke:

Eine Bank verteilt an Ihre Kunden Schlüssel und wählt dabei der Einfachheit halber immer das gleiche n und nur unterschiedliche e. Sie sendet die *gleiche* Nachricht verschlüsselt an mehrere Personen. Kann ein Horcher damit einen Angriff durchführen?

Genauer: m wird mit (n, e_1) und (n, e_2) verschlüsselt ($\gcd(e_1, e_2) = 1$):

$$c_1 = m^{e_1} \bmod n, \qquad c_2 = m^{e_2} \bmod n.$$

Kann man durch Kenntnis von n, e_1, e_2, c_1 und c_2 auf m schließen?

Antwort: Ja!

Der euklidische Algorithmus liefert x_1, x_2 mit $x_1 e_1 + x_2 e_2 = 1$

$$\Rightarrow \qquad {c_1}^{x_1} \cdot {c_2}^{x_2} = m^{e_1 x_1} \cdot m^{e_2 x_2} = m^{e_1 x_1 + e_2 x_2} = m \mod n.$$

Beispiel:

$n = 221\ (= 13 \cdot 17)$, $e_1 = 5$, $e_2 = 7$.
Abgehört wird $c_1 = 141$ und $c_2 = 45$. Was ist m?

Es ist $3 \cdot 5 + (-2) \cdot 7 = 1$, also

$$m = 141^3 \cdot 45^{(-2)} = 57 \cdot 43 = 20 \mod 221.$$

Test: $20^5 = 141 \bmod 221$, $20^7 = 45 \mod 221$.

Verallgemeinerung:

Das RSA-Verfahren kann man wie folgt verallgemeinern:

Sei G eine endliche Gruppe mit $|G|$ Elementen und e, d zwei Zahlen mit $ed = 1 \bmod |G|$ (Beim RSA-Verfahren ist $G = (\mathbb{Z}_{pq})^\times$).

Öffentlicher Schlüssel: G, e

Privater Schlüssel: d.

Ein $m \in G$ wird verschlüsselt zu

$$c = m^e := \underbrace{m \circ \ldots \circ m}_{e\text{-mal}}.$$

Die Entschlüsselung erfolgt durch

$$c^d = m^{ed} = m^{k \cdot |G| + 1} = \left(m^{|G|}\right)^k \circ m \overset{\text{Satz1.3}}{=} m.$$

Dazu muss es schwierig sein, zu G und e ein d mit $ed = 1 \bmod |G|$ zu berechnen. Dies ist einfach, sobald $|G|$ bekannt ist. Es muss also (ohne Zusatzwissen) schwierig sein, die Anzahl $|G|$ der Elemente von G zu bestimmen. Um e und d zu berechnen, muss es aber mit Zusatzwissen möglich sein, $|G|$ zu bestimmen. Bisher sind keine Gruppen bekannt, mit denen man dies in prinzipiell anderer Weise als beim RSA-Verfahren machen kann.

9 Hashfunktion

Definition:

Eine Hashfunktion ist eine Abbildung, die Texte beliebiger Länge auf eine Zeichenfolge fester Länge abbildet.

Eine Hashfunktion h heißt

Einwegfunktion genau dann, wenn es schwierig ist, zu gegebenem y_0 ein x_0 zu finden mit $h(x_0) = y_0$.

schwach kollisionsresistent genau dann, wenn es schwierig ist, zu einem gegebenen x ein $x' \neq x$ zu finden mit $h(x) = h(x')$.

stark kollisionsresistent genau dann, wenn es schwierig ist, x und x' zu finden mit $x' \neq x$ und $h(x) = h(x')$.

Eine kryptografische Hashfunktion ist eine stark kollisionsresistente Einweg-Hashfunktion.

Bekannte Hashfunktionen sind MD5 (veraltet), SHA-1 (gilt nicht mehr als kollisionsresistent), die SHA-2-Familie (SHA-224, SHA-256, SHA-384 und SHA-512) und SHA-3.

Anwendungen von Hashfunktionen:

Überprüfung öffentlicher Schlüssel: Den Hashwert seines öffentlichen Schlüssels kann man beispielsweise auf seine Visitenkarte drucken. Ein Kommunikationspartner, dem man eine Visitenkarte überreicht hat, kann so überprüfen, ob er den richtigen öffentlichen Schlüssel besitzt. Hier wird die schwache Kollisionsresistenz ausgenutzt.

Hinterlegung von Passwörtern: Statt Passwörter im Klartext zu speichern, wird der Hashwert der Passwörter gespeichert. Damit kann bei Ausspionieren der Datei noch nicht auf die Passwörter geschlossen werden (Einweg-Eigenschaft).

Datenintegrität: Zu einem Datensatz wird der Hashwert berechnet und gespeichert. Eine spätere Berechnung mit gleichem Ergebnis gewährleistet, dass die Daten in der Zwischenzeit nicht manipuliert wurden (schwache Kollisionsresistenz).

Einsatz bei Suchalgorithmen.

Die Geburtstagsattacke

Vorbetrachtung:

Eine Urne enthalte n nummerierte Kugeln. Wieviel Ziehungen (mit Zurücklegen) sind nötig, damit man mit einer Wahrscheinlichkeit ≥ 0.5 mindestens eine Kugel doppelt gezogen hat?

Antwort: Man benötigt ca. $1.2 \cdot \sqrt{n}$ Ziehungen.

Beispiel:

Bei $23 \approx 1,2 \cdot \sqrt{365}$ Personen in einem Raum ist die Wahrscheinlichkeit, dass zwei Personen am gleichen Tag Geburtstag haben, größer als 0.5.

Wegen der erstaunlich niedrigen Zahl nennt man dies das *Geburtstagsparadoxon.*

Bemerkung:

Die schwache Kollisionsresistenz entspricht dem Problem, jemanden im Raum zu finden, der an einem bestimmten Tag Geburtstag hat, die starke Kollisionsresistenz dem, zwei mit gleichem Geburtstag zu finden. Das erste ist schwerer zu realisieren als das zweite. Es ist leichter, etwas selten Vorkommendes zu verhindern als etwas häufiges. Daher die Namen „schwache“ und „starke“ Kollisionsresistenz.

MACs

Definition:

Eine schlüsselabhängige Hashfunktion heißt *Message Authentication Code* (*MAC*) oder *Message Integrity Code* (*MIC*).

Anwendung:

Ein MAC kann zur personenbezogenen Datenintegrität genutzt werden: Zwei Kommunikationspartner besitzen einen gemeinsamen Schlüssel k. Sendet der eine die Nachricht m mit dem MAC-Wert $h_k(m)$, so kann der Empfänger durch entsprechende Berechnung von $h_k(m)$ verifizieren, dass die Nachricht m tatsächlich vom Sender stammt.

Bemerkung:

HMAC (hashed message authentication code) ist eine Möglichkeit einen MAC aus einer Hashfunktion f zu konstruieren:

$$HMAC_k(m) = h((k \oplus a)||h((k \oplus b)||m)).$$

Dabei ist „$\oplus$“ die XOR-Verknüpfung, „$||$“ die Konkatenation und a und b zwei Konstanten. Die Konstruktion $h((k \oplus b)||m)$ allein wäre bei gewissen Hashfunktionen, die den Wert iterativ blockweise berechnen, nicht sicher, da ein Angreifer dann Blöcke an m anhängen, den Hashwert entsprechend weiterrechnen und so einen gültigen MAC zu einer Nachricht m' ohne Kenntnis des Schlüssels bilden könnte.

10 Digitale Signatur

Ziel der digitalen Signatur ist eine Analogie zur handschriftlichen Unterschrift: Man will zu einem Dokument eine „digitale Unterschrift“ erzeugen, die

1. nur der Unterschreiber leisten kann,
2. jeder nachprüfen kann.

Damit will man erreichen

dass der Empfänger sicher sein kann, von welchem Absender das Dokument kommt.

dass der Absender nicht abstreiten kann, das Dokument gesendet zu haben.

Die digitale Signatur ist in gewisser Weise die Umkehrung der Public-Key-Verschlüsselung:

Public-Key-Verschlüsselung:

Klartext
↓
Verschlüsselung mit öffentlicher Funktion $f_{\text{öff}}$
↓
Geheimtext
↓
Entschlüsselung mit privater Funktion f_{priv}
↓
Klartext

Es gilt $f_{\text{priv}} \circ f_{\text{öff}} = \text{id}$ und $f_{\text{öff}}$ ist ohne Zusatzwissen nicht umkehrbar.

Digitale Signatur:

Klartext
↓
Signieren mit privater Funktion f_{priv}
↓
Signatur
↓
Überprüfung mit öffentlicher Funktion $f_{\text{öff}}$

Die Überprüfung kann dadurch geschehen, dass $f_{\text{öff}} \circ f_{\text{priv}} = \text{id}$ ist. Damit die Signatur nicht gefälscht werden kann, muss es schwierig sein, zu einem Klartext m ein s zu finden mit $f_{\text{öff}}(s) = m$, d.h., $f_{\text{öff}}$ ist (ohne Zusatzwissen) nicht umkehrbar.

Die digitale Signatur ist wie ein gläserner Tresor: Jeder kann sich (mit $f_{\text{öff}}$) von der Richtigkeit der Signatur überzeugen, aber nur der Signierer kann ein Dokument (mit f_{priv}) in den Tresor legen.

RSA-Signatur (1):

Das RSA-Verfahren kann man direkt auf eine digitale Signatur übertragen:

Schlüsselerzeugung:

Wähle zwei (große) verschiedene Primzahlen p und q.

Setze $n = p \cdot q$ und $f = (p-1) \cdot (q-1) = \phi(n)$.

Wähle ein e mit $\gcd(e, f) = 1$.

Berechne $d = e^{-1} \bmod f$.

Öffentlicher Schlüssel: n, e

Privater Schlüssel: d.

Ver- und Entschlüsselung (Wiederholung):

Ein Klartext liege als Zahl $m \in \mathbb{Z}_n$ vor.

Verschlüsselung: $c = f_{\text{öff}}(m) = m^e \bmod n$.

Entschlüsselung: $m = f_{\text{priv}}(c) = c^d \bmod n$.

Die Entschlüsselung funktioniert, da für jedes $m \in \mathbb{Z}_n$ gilt (s. Kapitel 8):

$$f_{\text{priv}} \circ f_{\text{öff}}(m) = \left(m^e\right)^d \bmod n = m^{ed} \bmod n = m.$$

Umgekehrt gilt auch

$$f_{\text{öff}} \circ f_{\text{priv}}(m) = \left(m^d\right)^e \bmod n = m^{ed} \bmod n = m.$$

Man kann also wie folgt signieren:

Ein Klartext m liege vor.

Berechnung der Signatur (unter Zuhilfenahme des privaten Schlüssels d):

$$s := m^d \bmod n$$

Verifikation von (m, s) mit Hilfe des öffentlichen Schlüssels e:

Es muss $m = s^e \bmod n$ sein.

Existentielle Fälschung:

Eine existentielle Fälschung ist die Erzeugung eines Tupels (m, s) ohne Kenntnis des privaten Schlüssels, so dass s die Signatur zu m ist.

Bei der RSA-Signatur geht das leicht:

Wähle ein beliebiges s,

Berechne $m = s^e \bmod n$.

Dann ist s Signatur zu m, denn bei der Überprüfung wird $m = s^e \bmod n$ verifiziert.

Damit hat der Angreifer zwar keine Kontrolle über die „signierte“ Nachricht m, aber ist die zu signierende Nachricht beispielsweise ein Geldbetrag am Geldautomaten oder ein Schlüssel für ein symmetrisches Verschlüsselungsverfahren, so ist die existentielle Fälschung eine Gefahr

Statt langer Texte und um existentielle Fälschung zu verhindern signiert man nur den Hashwert einer Datei.

RSA-Signatur (2) mit Hashfunktion:

Ein Klartext m liege vor.

Berechnung der Signatur (unter Zuhilfenahme des privaten Schlüssels d und einer (öffentlichen) Hashfunktion h):

$$s := h(m)^d \bmod n$$

Verifikation von (m, s) mit dem öffentlichen Schlüssels e und der Hashfunktion h:

Es muss $h(m) = s^e \bmod n$ sein.

Bemerkungen:

1. Die existentielle Fälschung bei der RSA-Signatur mit Hashfunktion wird verhindert: Zu gewähltem s kann der Angreifer $h = s^e \bmod n$ berechnen. Wegen der Einweg-Eigenschaft von h kann er aber keine Nachricht m finden mit $h = h(m)$.
2. Durch die feste Länge von $h(m)$ braucht man keine Blockbildung oder Ähnliches bei der Signatur mit einer Hashfunktion.
3. Wenn nur der Hashwert h eines Dokuments signiert wird, sollte es zur Vermeidung des Übertragens einer Signatur von einem auf andere Texte schwierig sein, zu einem Hashwert ein anderes Dokument mit gleichem Hashwert zu finden. Dies wird durch die schwache Kollisionsresistenz von h gewährleistet.

   Es sollte sogar schwer sein, überhaupt zwei Dokumente mit gleichem Hashwert zu finden. Dies wird durch die starke Kollisionsresistenz von h gewährleistet.

Geburtstagsattacke:

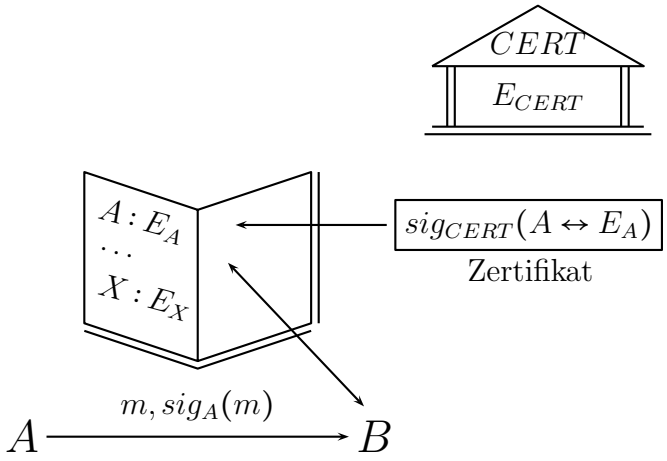
Beim unauthorisierten Entschlüsseln muss man zu einem Geheimtext c *den* Klartext m finden mit $f_{\text{öff}}(m) = c$. Gibt es n Klartexte, so braucht man bei Brute-Force im Durchschnitt $\frac{n}{2}$ Versuche, bei $n \approx 2^{128}$ also 2^{127} Versuche, was heutzutage praktisch unmöglich ist. Reicht $n \approx 2^{128}$ auch für die Signatur?

Ein Angreifer kann versuchen, eine Kollision einer „gutartigen“ Nachricht m_1 und einer „bösen“ Nachricht m_2, also $f(m_1) = f(m_2)$, zu finden, und legt dem Opfer dann m_1 zur Signatur vor. Die Signatur ist gleichzeitig Signatur von m_2.

Ähnlich wie beim Geburtstagsparadoxon sind bei n Hashwerten die Bildung von ca. $\sqrt{n}$ Werten nötig, um mit einer Wahrscheinlichkeit ≥ 0.5 eine Kollision zu finden. Sucht man diese Texte zur einen Hälfte „nahe“ bei m_1, zur anderen nahe bei m_2, so ist die Wahrscheinlichkeit einer Kollision guter und böser Nachricht gleich 0.5. Bei $n \approx 2^{128}$ ist der Aufwand $\sqrt{n} \approx 2^{64}$. Es ist problemlos möglich, zu einem Text 2^{64} auf den ersten Blick gleichlautende Texte zu bilden (z.B. durch Einfügen/Nicht-Einfügen von Leerzeichen an 64 verschiedenen Stellen).

Hierarchische Public-Key-Infrastruktur:

Ein Problem bei der Verifikation einer Signatur ist die Überprüfung, ob der öffentliche Schlüssel tatsächlich zum vermuteten Signierer gehört. Dies kann durch eine persönliche Übergabe oder aber durch eine offizielle Zertifizierungsstelle geschehen, indem diese den öffentlichen Schlüssel signiert.



Statt dass B in einem Verzeichnis den öffentlichen Schlüssel samt Signatur der Zertifizierungsstelle nachschlägt, sendet üblicherweise A mit der signierten Nachricht auch seinen öffentlichen Schlüssel und ein Zertifikat der Zertifizierungsstelle, dass der Schlüssel zu ihm gehört. Als Format für Zertifikate hat sich der *X.509*-Standard durchgesetzt.

Um ein Zertifikat zu überprüfen, benötigt B den öffentlichen Schlüssel der Zertifizierungsstelle. Einige Zertifizierungsanbieter lassen ihre Schlüssel direkt in Browser integrieren. Man kann die Schlüssel auch häufig im Internet finden, sollte aber auch

immer einen alternativen Weg der Überprüfung wählen, z.B. den Vergleich von Hashwerten mittels Telefon.

Eine solche *Public-Key-Infrastruktur* (*PKI*) kann z.B. auch firmenintern organisiert sein. Die Zertifizierungsstellen heißen auch *Certification Authorities* (*CA*), die oberste Instanz *Wurzelinstanz* oder *Root-CA*.

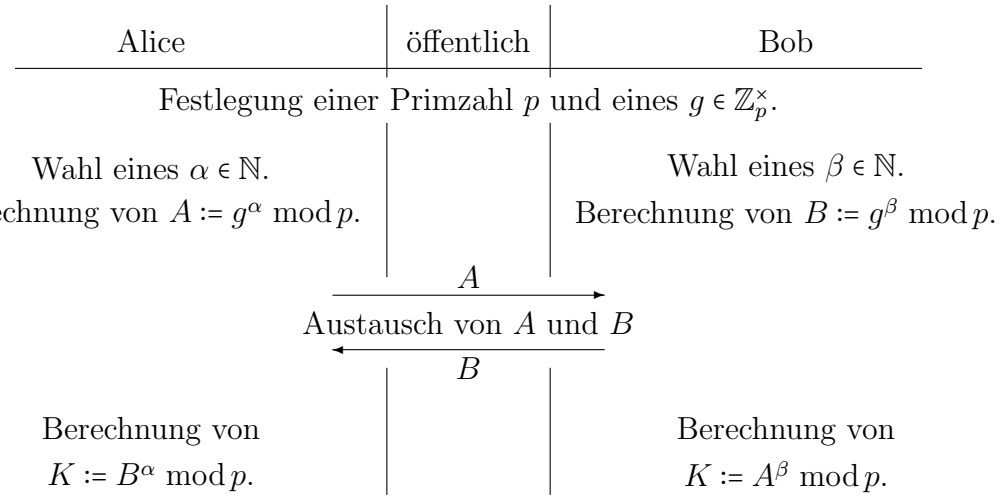
Gesetzliche Grundlage für digitale Signaturen in Deutschland sind das *Vertrauensdienstegesetz* (als Nachfolger des alten Signaturgesetzes) und die *EU-Verordnung 910/2014*, auch *eIDAS-Verordnung* genannt. In letzter wird definiert, was elektronische Signaturen, fortgeschrittene elektronische Signaturen und qualifizierte elektronische Signaturen sind. Qualifizierte elektronische Signaturen sind handschriftlichen Unterschriften gleichgestellt. Für eine qualifizierte elektronische Signaturen braucht man ein „qualifiziertes Zertifikat“ von einem „qualifizierten Vertrauensdiensteanbieter“. Das Vertrauensdienstegesetz regelt, dass in Deutschland das *Bundesamtes für Sicherheit in der Informationstechnik* im Webseiten-Bereich und die *Bundesnetzagentur* für alle anderen Belange entsprechend zuständig sind.

11 Diffie-Hellman-Schlüsselaustausch

Der Diffie-Hellman-Schlüsselaustausch ermöglicht zwei Kommunikationspartnern die Vereinbarung eines geheimen Schlüssels über einen öffentlichen Kanal, ohne vorher in Kontakt gekommen zu sein. Das Verfahren wird beispielsweise bei „https“ und „ssh“ benutzt.

Ablauf:

Alice und Bob wollen einen geheimen Schlüssel vereinbaren:



Tatsächlich ist

$$K = B^\alpha = \left(g^\beta\right)^\alpha = g^{\alpha\beta} = \left(g^\alpha\right)^\beta = A^\beta \bmod p.$$

Bemerkung:

Nach dem kleinen Satz von Fermat (Satz 7.2) ist $g^{p-1} = 1 \bmod p$, so dass sinnvollerweise $\alpha, \beta < p - 1$ sind.

Die Sicherheit des Verfahrens beruht auf der Schwierigkeit, aus der Kenntnis von $A := g^\alpha \bmod p$ bzw. $B := g^\beta \bmod p$ auf α bzw. β zu schließen. Dies ist besonders schwierig, wenn g^x für $x = 1, 2, \ldots, p - 1$ möglichst viele Werte durchläuft.

Beispiel:

In $\mathbb{Z}_{11}$ ist

x	1	2	3	4	5	6	7	8	9	10
2^x	2	4	8	5	10	9	7	3	6	1
3^x	3	9	5	4	1	3	9	5	4	1

Definition:

Sei p eine Primzahl. $g \in \mathbb{Z}_p$ heißt *Primitivwurzel* modulo p genau dann, wenn

$$\{g^x | x = 1, 2, \ldots, p - 1\} = \{1, 2, \ldots, p - 1\} = \mathbb{Z}_p^\times.$$

Ist g eine Primitivwurzel modulo p, so bezeichnet man die eindeutige Lösung $x \in \{1, \ldots, p-1\}$ der Gleichung $c = g^x \bmod p$ ($c \in \mathbb{Z}_p^\times$) mit $x = \log_g c$ (*diskreter Logarithmus* von c zur Basis g).

Beispiel:

2 ist Primitivwurzel modulo 11, 3 nicht.
In $\mathbb{Z}_{11}$ ist $\log_2 5 = 4$.

Satz 11.1:

Zu jeder Primzahl p gibt es eine Primitivwurzel modulo p.

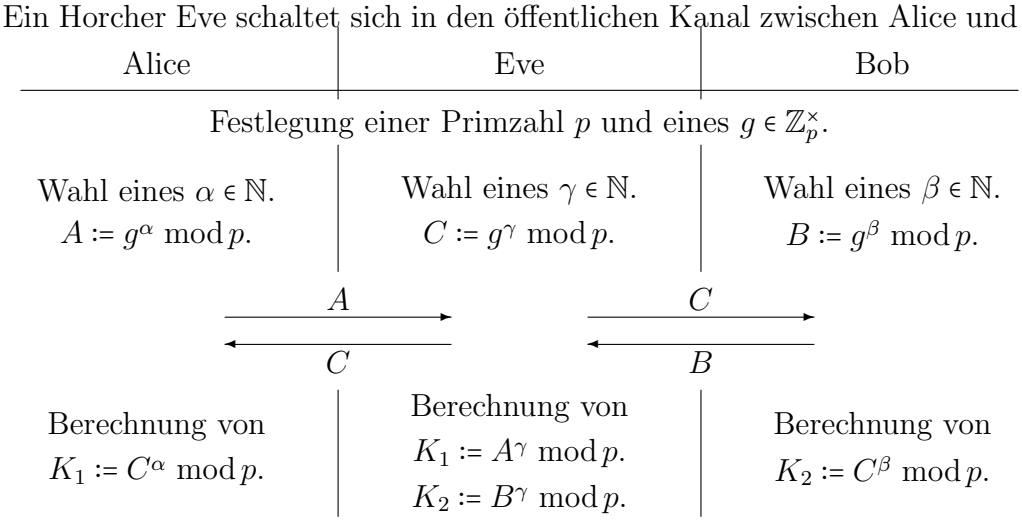
Bemerkung:

Wer diskrete Logarithmen effizient berechnen kann, der kann den Diffie-Hellman-Schlüsselaustausch brechen, denn er kann aus A, B und p dann α und β und damit den geheimen Schlüssel $K = g^{\alpha\beta}$ errechnen. Bis heute (Stand Herbst 2023) ist dazu allerdings kein Algorithmus bekannt. Es ist bisher auch nicht bekannt, ob es ohne Berechnung diskreter Logarithmen möglich ist, aus A, B und p den geheimen Schlüssel K zu errechnen.

Hat p eine Länge von ca. 2048 Bits, so wird der Diffie-Hellman-Schlüsselaustausch zur Zeit als sicher angesehen.

Angriffsmöglichkeit (Man-in-the-middle-Attacke):

Ein Horcher Eve schaltet sich in den öffentlichen Kanal zwischen Alice und Bob ein:



Nun hat Eve je einen Schlüssel mit Alice und Bob:

$$K_1 = C^\alpha = g^{\alpha\gamma} = A^\gamma \bmod p \quad \text{und} \quad K_2 = C^\beta = g^{\beta\gamma} = B^\gamma \bmod p.$$

Eve kann damit alle Nachrichten zwischen Alice und Bob mitlesen:

Schickt Alice eine mit K_1 verschlüsselte Nachricht an Bob, fängt Eve die Nachricht ab, entschlüsselt sie mit K_1, kann sie lesen, verschlüsselt sie mit K_2 und leitet sie weiter an Bob.

Verallgemeinerung:

Der Diffie-Hellman-Schlüsselaustausch kann auf eine beliebige Gruppe $(G, \circ)$ verallgemeinert werden:

Statt einer Primzahl p und $g \in \mathbb{Z}_p^\times$ legt man eine Gruppe $(G, \circ)$ und ein $g \in G$ fest. Mit der Schreibweise

$$g^n := \underbrace{g \circ \ldots \circ g}_{n\text{-mal}}$$

lässt sich das Verfahren dann weiter wörtlich wie oben durchführen.

Das Verfahren ist sicher, wenn die Berechnung des Logarithmus in G, d.h. das Auflösen von $g^x = c$ nach x, schwierig ist. Besonders schwierig wird dies, wenn g^x sämtliche Gruppenelemente durchläuft.

Bemerkungen:

1. Auch bei allgemeinen endlichen Gruppen nennt man die Lösung x der Gleichung $g^x = c$ diskreter Logarithmus.
2. In der Praxis werden neben $(\mathbb{Z}_p^\times, \cdot)$ als Gruppen auch sogenannte elliptische Kurven verwendet.

12 Elgamal

Elgamal ist neben RSA die verbreitetste Realisierung eines Public-Key-Verfahrens. Es ist benannt nach seinem Entdecker, dem Ägypter Taher Elgamal.

Schlüsselerzeugung:

Wähle eine (große) Primzahl p und ein $g \in \mathbb{Z}_p^\times$ (möglichst eine Primitivwurzel).

Wähle ein $\alpha \in \{1, \ldots, p-1\}$ und berechne $A := g^\alpha \bmod p$.

Öffentlicher Schlüssel: p, g, A

Privater Schlüssel: α.

Ver- und Entschlüsselung:

Ein Klartext liege als Zahl $m \in \mathbb{Z}_p$ vor.

Verschlüsselung:

Wähle eine Zahl $\beta \in \{1, \ldots, p-1\}$ und berechne

$$B := g^\beta \bmod p \qquad \text{und} \qquad c := A^\beta \cdot m \bmod p.$$

Der verschlüsselte Klartext ist das Tupel (B, c).

Entschlüsselung:

Berechne $K := B^\alpha \bmod p$ und K^{-1} in $\mathbb{Z}_p$. Der Klartext ergibt sich dann als

$$m = K^{-1} \cdot c \bmod p,$$

denn es ist

$$K = B^\alpha = \left(g^\beta\right)^\alpha = \left(g^\alpha\right)^\beta = A^\beta \bmod p$$

und daher

$$K^{-1} \cdot c = K^{-1} \cdot A^\beta \cdot m = K^{-1} \cdot K \cdot m = m \bmod p.$$

Bemerkungen:

1. Wer diskrete Logarithmen bestimmen kann, der kann das Elgamal-Verfahren brechen. Er kann dann aus $A = g^\alpha \bmod p$ die Zahl α errechnen und ist so im Besitz des privaten Schlüssels und kann alle Texte wie oben entschlüsseln.

   Es ist bisher allerdings nicht bekannt, ob jemand, der das Elgamal-Verfahren brechen kann, auch diskrete Logarithmen berechnen kann.

   Bei einer Schlüssellänge von 2048 Bits (Länge von p) wird das Elgamal-Verfahren zur Zeit als sicher angesehen.

2. Angriffsmöglichkeit bei gleichem β:

   Verwendet ein Sender immer das gleiche β, so kann ein Angreifer bei Kenntnis eines Klartext-Geheimtext-Paares (m_0, c_0) jeden weiteren Geheimtext c entschlüsseln. Aus m_0 und $c_0 = A^\beta \cdot m_0 \bmod p$ kann er nämlich $K = A^\beta \bmod p = {m_0}^{-1} \cdot c_0 \bmod p$ errechnen und damit jeden weiteren Geheimtext $c = A^\beta \cdot m \bmod p$ entschlüsseln zu $m = K^{-1} \cdot c \bmod p$. Zur Verschlüsselung muss man daher immer neue Zahlen β generieren.

Verallgemeinerung:

Ähnlich wie beim Diffie-Hellman-Schlüsselaustausch kann man das ElGamal-Verfahren verallgemeinern, indem man statt $\mathbb{Z}_p^\times$ eine beliebige Gruppe G wählt und ein Element $g \in G$ auszeichnet (möglichst einen Erzeuger der Gruppe). Schlüsselerzeugung, Ver- und Entschlüsselung eines Klartextes $m \in G$ gehen dann analog zu oben.

In der Praxis werden neben $(\mathbb{Z}_p^\times, \cdot)$ als Gruppen auch sogenannte elliptische Kurven verwendet.

Elgamal-Signatur

Ein zur Elgamal-Verschlüsselung ähnliches Verfahren kann zur Signatur benutzt werden:

Schlüsselerzeugung (wie oben):

Wähle eine (große) Primzahlen p und ein $g \in \mathbb{Z}_p^\times$ (möglichst eine Primitivwurzel).

Wähle ein $\alpha \in \{1, \dots, p-1\}$ und berechne $A := g^\alpha \bmod p$.

Öffentlicher Schlüssel: p, g, A

Privater Schlüssel: α.

Signierung:

Ein Klartext liege als Zahl $m \in \{1, \dots, p-1\}$ vor.

Wähle eine Zufallszahl $k \in \{1, \dots, p-1\}$, die zu $p-1$ teilerfremd ist.

Berechne k^{-1} modulo $p-1$, d.h. eine Lösung x zu

$$k \cdot x = 1 \bmod (p-1).$$

Setze $r := g^k \bmod p$, $s := k^{-1}(m - \alpha \cdot r) \bmod (p-1)$.

Die Signatur zu m ist (r, s).

Verifikation:

Prüfe, ob

$$A^r r^s = g^m \bmod p \qquad (1)$$

erfüllt ist. Falls ja, so ist die Signatur in Ordnung, falls nein, nicht.

Warum gilt (1)?:

Es ist

$$A^r r^s = (g^\alpha)^r (g^k)^s = g^{\alpha r + ks} \bmod p.$$

Nun ist

$$\alpha r + ks = \alpha r + k \cdot k^{-1}(m - \alpha \cdot r) = \alpha r + m - \alpha r = m \bmod (p-1),$$

also $\alpha r + ks = m + l \cdot (p-1)$ mit einem $l \in \mathbb{Z}$. Wegen $g^{p-1} = 1 \bmod p$ (kleiner Satz von Fermat, Satz 7.2) folgt

$$A^r r^s = g^{\alpha r + ks} = g^{m + l \cdot (p-1)} = g^m \cdot (g^{p-1})^l = g^m \bmod p.$$

Bemerkungen:

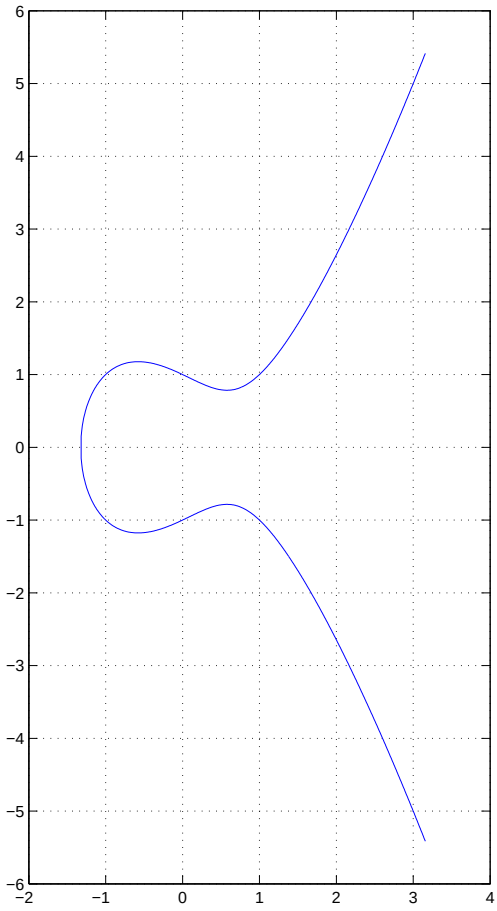
1. In einer etwas abgeänderten (effektiveren) Version ist das Elgamal-Verfahren als *Digital Signature Algorithm* (DSA) standardisiert.
2. Wie bei der Verschlüsselung, kann man die Elgamal-Signatur auf beliebige Gruppen verallgemeinern, z.B. auf elliptische Kurven.
3. Man kann allein aus Kenntnis des öffentlichen Schlüssels ein m und eine dazu gültige Signatur (r, s) konstruieren (*existentielle Fälschung*). Man kann das m zwar nicht vorgeben (es ergibt sich aus der Rechnung), aber wie bei der RSA-Signatur ist das eine Schwäche. Um diese Schwäche zu umgehen signiert man wie bei RSA nie die eigentliche Nachricht sondern immer den Hashwert einer Nachricht. Durch den Angriff kann dann ein Hashwert und eine dazu gültige Signatur erzeugt werden, aber bei einer kryptografisch sicheren Hashfunktion ist es (praktisch) unmöglich, eine passende Nachricht zu dem Hashwert zu finden.

13 Elliptische Kurven

Beispiel:

Betrachte in $\mathbb{R}^2$ die Punktmenge

$$k := \{(x, y) | y^2 = x^3 - x + 1\}.$$



Sei g die Gerade durch $(0, -1)$ und $(1, 1)$. g schneidet die Kurve noch in einem weiteren Punkt: Die Geradengleichung zu g ist $y = 2x - 1$. Für Schnittpunkte von g und k gilt also

$$\begin{aligned} & (2x-1)^2 = x^3 - x + 1 \\ \Leftrightarrow \quad & 4x^2 - 4x + 1 = x^3 - x + 1 \\ \Leftrightarrow \quad & 0 = x^3 - 4x^2 + 3x. \end{aligned}$$

Nach Konstruktion sind $x = 0$ und $x = 1$ Lösungen dieser kubischen Gleichung. Durch Polynomdivision durch $(x - 0)$ und $(x - 1)$ erhält man einen linearen Term, der eine dritte Nullstelle liefert:

$$(x^3 - 4x^2 + 3x) : [(x - 0) \cdot (x - 1)] = x - 3.$$

Also ist $x = 3$ und dazu $y = 2 \cdot 3 - 1 = 5$ der dritte Schnittpunkt.

Entsprechend hat jede Gerade, die K in zwei verschiedenen Punkten schneidet, einen dritten Schnittpunkt. Dabei zählen Berührpunkte als doppelte Schnittpunkte, z.B. bei der Geraden durch $(-1,-1)$ und $(1,1)$: Die Gerade berührt k in $(1,1)$. Dieser Punkt zählt doppelt.

Geraden, die parallel zur y-Achse verlaufen, bilden eine Ausnahme. Hier kann man „unendlich“ als dritten Schnittpunkt ansehen.

Man kann auf Kurven dieser Form nun eine Verknüpfung definieren, indem man zwei Punkten den dritten Schnittpunkt der Geraden durch diese Punkte zuordnet. Tatsächlich erweist es sich als günstig, bei der Zuordnung nicht diesen dritten Punkt sondern dessen Spiegelpunkt bei Spiegelung an der x-Achse zu nehmen:

Definition:

Sei $4a^3 + 27b^2 \neq 0$ und $k := \{(x,y) | y^2 = x^3 + ax + b\}$.

$K := k \cup \{\mathcal{O}\}$ heißt *elliptische Kurve*.

Auf K wird wie folgt eine Verknüpfung definiert:

Zu $P_1 = (x_1, y_1)$, $P_2 = (x_2, y_2)$ aus k sei

$$P_1 + P_2 := \begin{cases} \mathcal{O} & \text{, falls } x_1 = x_2 \text{ und } y_1 = -y_2, \\ (x_3, y_3) & \text{, sonst,} \end{cases}$$

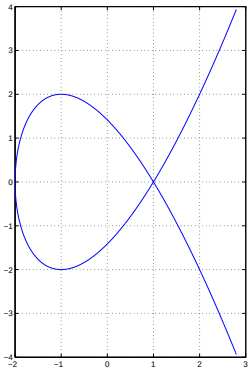
mit $x_3 = \lambda^2 - x_1 - x_2$ und $y_3 = \lambda(x_1 - x_3) - y_1$, wobei

$$\lambda := \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{, falls } P_1 \neq P_2, \\ \frac{3{x_1}^2 + a}{2y_1} & \text{, falls } P_1 = P_2. \end{cases}$$

Ferner: $P + \mathcal{O} := \mathcal{O} + P := P$.

Bemerkung:

Die Bedingung $4a^3 + 27b^2 \neq 0$ verhindert eine Singularität, wie sie z.B. bei $y^2 = x^3 - 3x + 2$ auftreten würde.



Beispiel (Fortsetzung):

Zu $P_1 = (0,-1)$ und $P_2 = (1,1)$ ist $\lambda = \frac{1-(-1)}{1-0} = 2$, also

$$x_3 = 2^2 - 0 - 1 = 3 \quad \text{und} \quad y_3 = 2 \cdot (0-3) - (-1) = -5:$$

$P_1 + P_2 = (3,-5)$.

Zu $P_1 = P_2 = (1,1)$ ist $\lambda = \frac{3 \cdot 1^1 + (-1)}{2 \cdot 1} = 1$, also

$$x_3 = 1^2 - 1 - 1 = -1 \quad \text{und} \quad y_3 = 1 \cdot (1-(-1)) - 1 = 1.$$

Durch elementares Nachrechnen kann man zeigen, dass „+“ auf K kommutativ und assoziativ ist. $\mathcal{O}$ ist bzgl. „+“ neutrales Element und für einen Punkt P gilt mit dem gespiegelten Punkt $\bar{P}$: $P + \bar{P} = \mathcal{O}$.

Satz 13.1

Eine elliptische Kurve mit „+“ ist eine kommutative Gruppe.

Beispiel für die Assoziativität:

$$\begin{aligned} \big[(0,-1) + (1,1)\big] + (0,1) &= (3,-5) + (0,1) = (1,1), \\ (0,-1) + \big[(1,1) + (0,1)\big] &= (0,-1) + (-1,-1) = (1,1). \end{aligned}$$

Bei der Definition der Verknüpfung werden nur elementare Rechenoperationen benutzt: Addition, Subtraktion (= Addition mit dem „+“-Inversen), Multiplikation und Division (= Multiplikation mit dem „$\cdot$“-Inversen). Man kann elliptische Kurven also über beliebigen Körpern bilden.

Beispiel:

Betrachte $K = \{(x,y) | y^2 = x^3 - x + 1\} \cup \{\mathcal{O}\}$ über $\mathbb{Z}_5$. Es ist

$$K = \{\mathcal{O},\ (0,4),\ (1,1),\ (3,0),\ (0,1),\ (1,4),\ (4,1),\ (4,4)\}.$$

Berechnung von $(1,1) + (3,0)$:

In $\mathbb{Z}_5$ ist $(3-1)^{-1} = 2^{-1} = 3$, also

$$\lambda = \frac{0-1}{3-1} = (-1) \cdot (3-1)^{-1} = 4 \cdot 3 = 2 \bmod 5,$$

also

$$x_3 = 2^2 - 1 - 3 = 0 \bmod 5 \quad \text{und} \quad y_3 = 2 \cdot (1-0) - 1 = 1 \bmod 5,$$

also $(1,1) + (3,0) = (0,1)$.

Elliptische Kurven über $\mathbb{Z}_p$, p Primzahl, werden als Gruppen für das ElGamal-Verfahren benutzt (ECC = *Elliptic curve cryptography*). Die Bestimmung diskreter Logarithmen ist hier (nach heutigem Stand) besonders schwer. Um eine Sicherheit entsprechend einem RSA-Verfahren mit RSA-Modul der Länge 2048-Bit zu erreichen, genügt bei ECC ein p mit ca. 256 Bit. Die kürzere Länge gleicht den Mehraufwand in der Arithmetik aus und bietet weitere Vorteile, z.B. auf Smart-Cards mit beschränktem Prozessor und/oder Speicherplatz.

14 Angriffsmethoden

1. Das quadratische Sieb

Ziel:

Finde einen echten Teiler von n.

Idee:

Finde x, y mit $x^2 = y^2 \bmod n$ und $x \neq \pm y \bmod n$. Dann gilt:

n teilt $x^2 - y^2 = (x-y)\cdot(x+y)$ aber weder $x+y$ noch $x-y$.

$\Rightarrow$ $\gcd(n, x-y)$ ist echter Teiler von n.

Suche von x, y:

Sei $m \approx \sqrt{n}$ und $f(x) := (x+m)^2 - n$.

Faktorisiere $f(x)$ für kleine x und suche Kombinationen mit im Produkt geraden Primzahlpotenzen. Beispiel:

$n = 7429$, also $m = 86$

$$
\begin{array}{rrrrrrrrr}
-1 & & & \cdot 373 & = & -373 & = & f(-2) & = & 84^2 \bmod n \\
-1 & \cdot 2^2 & \cdot 3 & \cdot 17 & = & -204 & = & f(-1) & = & 85^2 \bmod n \\
-1 & \cdot 3 & \cdot 11 & & = & -33 & = & f(0) & = & 86^2 \bmod n \\
& 2^2 & \cdot 5 & \cdot 7 & = & 140 & = & f(1) & = & 87^2 \bmod n \\
& & 3^2 & \cdot 5 \cdot 7 & = & 315 & = & f(2) & = & 88^2 \bmod n \\
& 2^2 & \cdot 3 & \cdot 41 & = & 492 & = & f(3) & = & 89^2 \bmod n
\end{array}
$$

$\Rightarrow$ $2^2 \cdot 3^2 \cdot 5^2 \cdot 7^2 = 87^2 \cdot 88^2 \bmod n$

$\Rightarrow$ $210^2 = 7656^2 = 227^2 \bmod n$

$\Rightarrow$ $\gcd(210 - 227, n) = 17$ ist ein echter Teiler von n.

Verbesserung der Faktorisierung:

Wegen

$$
\begin{aligned}
f(x+p) &= (x+p+m)^2 - n \\
&= (x+m)^2 + 2(x+m)p + p^2 - n \\
&= f(x) + p\cdot\big(2(x+m)+p\big)
\end{aligned}
$$

gilt

$$p | f(x) \Leftrightarrow p | f(x+p).$$

Damit kann man Probedivisionen sparen.

2. Shanks Babystep-Giantstep-Algorithmus

Ziel:

Finde zu A, g, p ein x mit $A = g^x \bmod p$.

Brute-Force:

Teste $x = 1, 2, \ldots p-2$.
Aufwand im Durchschnitt $\approx \frac{p}{2}$.

Idee:

Sei $w \approx \sqrt{p}$, $x = k \cdot w + l$

$$\Rightarrow \quad A = g^x = g^{k \cdot w + l} = (g^w)^k \cdot g^l$$
$$\Rightarrow \quad (g^w)^k = A \cdot g^{-l} = A \cdot (g^{-1})^l.$$

Algorithmus:

1. Berechne und speichere $A \cdot (g^{-1})^l$, $l = 0, 1, \ldots, w-1$ (Babysteps).
2. Berechne $(g^w)^k$, $k = 0, 1, \ldots$ und teste, ob das Ergebnis mit einem bei 1. berechneten Wert übereinstimmt ($\rightarrow$ Hashtabelle).

Beispiel:

Gesucht: x mit $5 = 15^x \bmod 101$, also $p = 101$, $w = 10$.
$15^{-1} \bmod 101 = 27$.

l	0	1	2	3	4	5	6	7	8	9
$5 \cdot 27^l \bmod 101$	5	34	9	41	97	94	13	48	84	46

$15^{10} \bmod 101 = 17$.

k	0	1	2	3	4	5	6	7	8	9
$17^k \bmod 101$	1	17	87	65	95	100	84			

$\Rightarrow \quad x = 6 \cdot 10 + 8 = 68.$

Aufwand:

Ca. $2 \cdot w \approx 2 \cdot \sqrt{p}$ Berechnungen plus Vergleiche.

15 Anhang: Der chinesische Restsatz

Problem:

Suche eine Lösung x zu

$$\begin{aligned} x &= a_1 \bmod m_1, \\ x &= a_2 \bmod m_2, \\ x &= a_3 \bmod m_3 \end{aligned}$$

mit teilerfremden m_1, m_2 und m_3.

Lösung:

Suche zunächst Lösungen x_1, x_2 und x_3 mit

$$\begin{aligned} &x_1 = 1 \bmod m_1, \quad x_2 = 0 \bmod m_1, \quad x_3 = 0 \bmod m_1, \\ &x_1 = 0 \bmod m_2, \quad x_2 = 1 \bmod m_2, \quad x_3 = 0 \bmod m_2, \\ &x_1 = 0 \bmod m_3, \quad x_2 = 0 \bmod m_3, \quad x_3 = 1 \bmod m_3. \end{aligned}$$

Hat man solche x_1, x_2 und x_3 gefunden, so ergibt sich x durch

$$x = a_1x_1 + a_2x_2 + a_3x_3.$$

Bestimmung von x_1 (x_2 und x_3 entsprechend):

x_1 ist ein Vielfaches von m_2 und m_3, also

$$x_1 = k \cdot m_2 \cdot m_3$$

(beachte: m_2 und m_3 sind teilerfremd). Aus $x_1 = 1 \bmod m_1$ folgt dann

$$\begin{aligned} & k \cdot m_2 \cdot m_3 = 1 \bmod m_1 \\ \Rightarrow \quad & k = (m_2 \cdot m_3)^{-1} \bmod m_1. \end{aligned}$$

Da m_1 teilerfremd zu $m_2 \cdot m_3$ ist, gibt es ein derartiges k (Bestimmung z.B. mittels des erweiterten euklidischen Algorithmus).

Beispiel:

Suche eine Lösung x zu

$$\begin{aligned} x &= 2 \bmod 3, \\ x &= 0 \bmod 4, \\ x &= 1 \bmod 5. \end{aligned}$$

Bestimmung von x_1 mit

$$\begin{aligned} x_1 &= 1 \bmod 3, \\ x_1 &= 0 \bmod 4, \\ x_1 &= 0 \bmod 5. \end{aligned}$$

Es ist

$$(4 \cdot 5)^{-1} \bmod 3 = 20^{-1} \bmod 3 = 2^{-1} \bmod 3 = 2,$$

also

$$x_1 = 2 \cdot 4 \cdot 5 = 40.$$

Entsprechend ergibt sich wegen

$$(3 \cdot 5)^{-1} \bmod 4 = 15^{-1} \bmod 4 = 3^{-1} \bmod 4 = 3$$

und

$$(3 \cdot 4)^{-1} \bmod 5 = 12^{-1} \bmod 5 = 2^{-1} \bmod 5 = 3,$$

dass

$$x_2 = 3 \cdot 3 \cdot 5 = 45 \quad \text{und} \quad x_3 = 3 \cdot 3 \cdot 4 = 36.$$

Damit ergibt sich als eine Lösung

$$x = 2 \cdot 40 + 0 \cdot 45 + 1 \cdot 36 = 116.$$

Frage:

Gibt es weitere Lösungen?

Antwort:

Ja!

Im Beispiel ist wegen $3 \cdot 4 \cdot 5 = 60$ z.B. jedes x mit $x = 116 \bmod 60$ auch eine Lösung, z.B. $x = 56$.

Frage:

Gibt es modulo $m_1 \cdot m_2 \cdot m_3$ weitere Lösungen?

Antwort:

Nein!

Begründung:

Zu jedem Tupel $(a_1, a_2, a_3) \in \mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \mathbb{Z}_{m_3}$ gibt es eine Lösung $x \in \mathbb{Z}_M$, $M := m_1 \cdot m_2 \cdot m_3$.
$\mathbb{Z}_M$ enthält M Elemente, $\mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \mathbb{Z}_{m_3}$ enthält $m_1 \cdot m_2 \cdot m_3 = M$ Elemente. Zu jedem Problem kann es daher nur *eine* Lösung in $\mathbb{Z}_M$ geben.

Ist x Lösung zu (a_1, a_2, a_3) und y Lösung zu (b_1, b_2, b_3), so gilt:

$$\begin{aligned} x + y &\quad \text{ist Lösung zu} \quad (a_1 + b_1, a_2 + b_2, a_3 + b_3) \\ x \cdot y &\quad \text{ist Lösung zu} \quad (a_1 \cdot b_1, a_2 \cdot b_2, a_3 \cdot b_3). \end{aligned}$$

Man sagt, $\mathbb{Z}_M$ und $\mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \mathbb{Z}_{m_3}$ sind *isomorph*. Statt in $\mathbb{Z}_M$ kann man auch in $\mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \mathbb{Z}_{m_3}$ rechnen.

Satz 15.1(Chinesischer Restsatz)

Seien $m_1, m_2, \ldots, m_k$ paarweise teilerfremd, $M := m_1 \cdot m_2 \cdot \ldots \cdot m_k$. Dann gibt es zu beliebigen $a_1, a_2, \ldots, a_k$ ein x mit

$$x = a_i \bmod m_i \qquad (i = 1, 2, \ldots k).$$

x ist modulo M eindeutig.
Die entsprechende Zuordnung definiert einen Isomorphismus zwischen $\mathbb{Z}_M$ und $\mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \ldots \times \mathbb{Z}_{m_k}$.

Aufgabe:

Seien $p, q > 3$ zwei verschiede Primzahlen und $n = pq$. Wieviel verschiedene Quadratwurzeln kann es zu einer Zahl modulo n geben?

Lösung:

Sei c eine Quadratzahl modulo n und a eine Quadratwurzel, also $a^2 = c \bmod n$. Sei $a_p = a \bmod p$, $a_q = a \bmod q$. Nach dem chinesischen Restsatz (Satz 15.1) gibt es a_1 bis a_4 mit

$$\begin{aligned} a_1 &= +a_p \bmod p \qquad \text{und} \qquad a_1 = +a_q \bmod q, \\ a_2 &= -a_p \bmod p \qquad \text{und} \qquad a_2 = +a_q \bmod q, \\ a_3 &= +a_p \bmod p \qquad \text{und} \qquad a_3 = -a_q \bmod q, \\ a_4 &= -a_p \bmod p \qquad \text{und} \qquad a_4 = -a_q \bmod q. \end{aligned}$$

Dann gilt

$$a_i{}^2 = a_p{}^2 = a^2 = c \bmod p \qquad \text{und} \qquad a_i{}^2 = a_q{}^2 = a^2 = c \bmod q.$$

Also ist $a_i{}^2 = c \bmod n$.

Die a_i sind unterschiedlich, da $+a_p \neq -a_p \bmod p$ gilt (sonst wäre $2a_p = 0 \bmod p$ im Widerspruch zu p Primzahl > 3), entsprechend für q. Damit gibt es vier Quadratwurzeln.

Weitere Quadratwurzeln gibt es nicht, denn gilt $b^2 = c \bmod n$, so folgt $b^2 = c = a^2 = a_p{}^2 \bmod p$, also

$$0 = b^2 - a_p{}^2 = (b - a_p) \cdot (b + a_p) \bmod p.$$

Da $\mathbb{Z}_p$ als Körper Nullteilerfrei ist, folgt $b = a_p$ oder $b = -a_p$, entsprechend für q, so dass b mit einem der a_i übereinstimmt.

Nochmals - Warum ist beim RSA-Verfahren $m = c^d \bmod n$?:

Erinnerung:

$n = p \cdot q$ mit Primzahlen p und q, $f = \phi(n) = (p-1)\cdot(q-1)$. Zu e wird ein d bestimmt mit $ed = 1 \bmod f$. $m \in \mathbb{Z}_n$ wird verschlüsselt zu $c = m^e \bmod n$. Behauptung: $m = c^d \bmod n$.

Es gilt

$$ed = k \cdot f + 1 = k \cdot (p-1) \cdot (q-1) + 1.$$

Falls $\gcd(m, p) = 1$ ist, gilt nach dem kleinen Satz von Fermat (Satz 7.2)

$$m^{ed} = \left(m^{p-1}\right)^{k(q-1)} \cdot m = 1^{k(q-1)} \cdot m = m \bmod p.$$

Falls $\gcd(m, p) \neq 1$ ist, folgt $p|m$, also $m^{ed} = 0 = m \bmod p$.

In jedem Fall gilt also $m^{ed} = m \bmod p$. Entsprechend gilt $m^{ed} = m \bmod q$ für alle m. Nach dem chinesischen Restsatz (Satz 15.1) folgt $m^{ed} = m \bmod p \cdot q$, also $m = c^d \bmod n = m$.