Security

- LV 4121 und 4241 -

Algebraische Strukturen und elementare Zahlentheorie

- Terminologie und Grundsätze der elementaren Zahlentheorie
- Herausstellung einer algebraischen Struktur
- Rechenregeln der modularen Arithmetik
- Zahlentheoretische Funktionen und Algorithmen
- Primzahlen und Primfaktorzerlegung
- Vertraulichkeitsschutz mittels einer Stromchiffre

Kap. 4: Algebraische Strukturen und elementare Zahlentheorie

Teil 1: Algebraische Strukturen und modulare Arithmetik

- Nomenklatur
- Algebraische Strukturen
  - Monoid
  - Gruppe
  - Ringe und Körper
- Modulare Arithmetik
  - Ganzzahlige Division (div und mod)
  - Teilbarkeit
  - Kongruenzen

N Menge der nichtnegativen ganzen Zahlen
$\mathbf{N} := \{0, 1, 2, 3, \cdots\}$

N+1 Menge der natürlichen (positiven ganzen) Zahlen
$\mathbf{N+1} := \{1, 2, 3, \cdots\} = \mathbf{N} \setminus \{0\}$

Z Menge der ganzen Zahlen
$\mathbf{Z} := \{0, \pm 1, \pm 2, \pm 3, \cdots\}$

P Menge der Primzahlen
$\mathbf{P} := \{2, 3, 5, 7, 11, 13, \cdots\}$

Q Menge der rationalen Zahlen (abb. oder period. nicht abb. Dezimalbrüche)
$\mathbf{Q} := \{a / b \mid a, b \in \mathbf{Z}, b \neq 0, ggT(a, b) = 1 \text{ (teilerfremd)}\}$

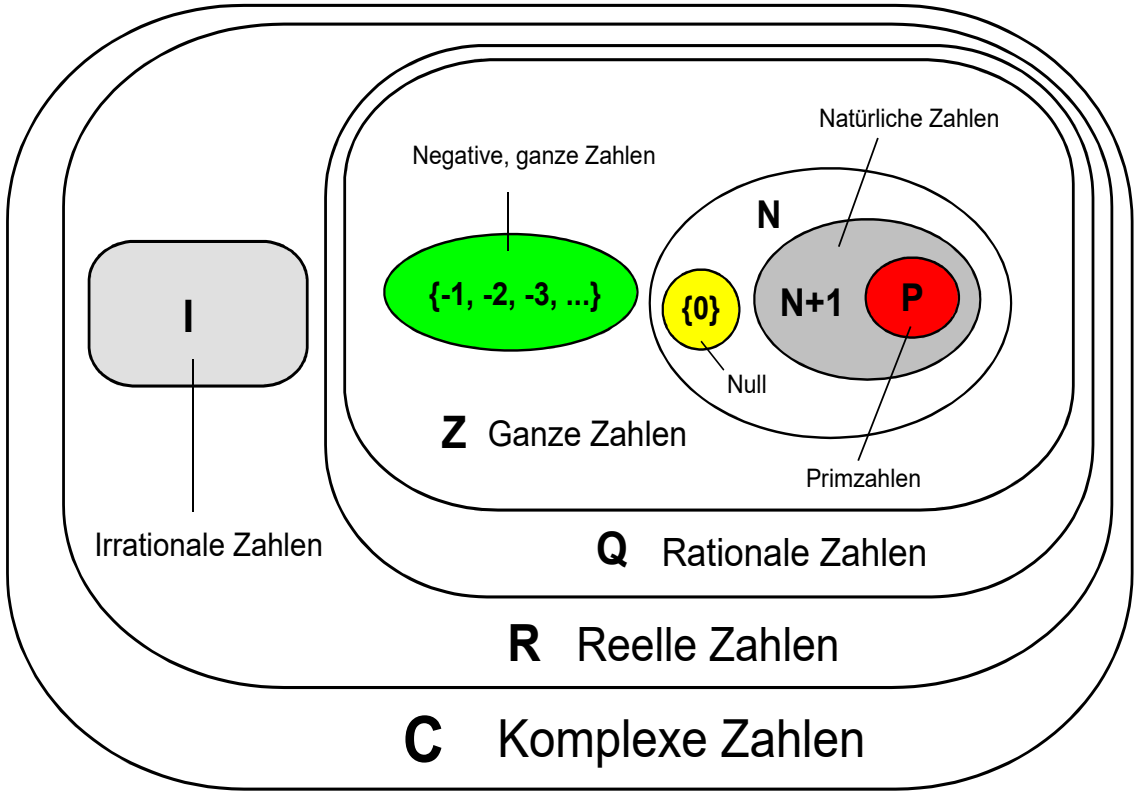
I Menge der irrationalen Zahlen (nicht-period. nicht abb. Dezimalbrüche)
$\mathbf{I} := \{\sqrt[n]{p} \mid p \in \mathbf{P};\ n = 2, 3, 4, \cdots ;\ \sin(\pi/4) ;\ e ;\ \pi ;\ \cdots\}$

R Menge der reellen Zahlen
$\mathbf{R} := \mathbf{Q} \cup \mathbf{I}$

C Menge der komplexen Zahlen
$\mathbf{C} := \{a + j\,b \mid a, b \in \mathbf{R} ;\ j^2 = -1\}$

$\mathbf{Z_m}$ Restklassenring modulo m
$\mathbf{Z_m} := \{0, 1, 2, 3, \cdots, m - 1\}$

Es gelten die Zusammenhänge: $\mathbf{N \subset Z \subset Q \subset R \subset C}$ **und** $\mathbf{R = Q \cup I}$



$$\mathbf{[k:m]} := \begin{cases} 0\ , \text{ falls } k > m \\ \{z \mid z \in \mathbf{Z}\ , k \le z \le m\}\ , \text{ sonst} \end{cases} \qquad k, m \in \mathbf{Z}$$

n | k n teilt k

$\Rightarrow (\exists\ x \in \mathbf{Z})\ \ k = x \cdot n$

n ∤ k n teilt k nicht

$\Rightarrow (\forall\ x \in \mathbf{Z})\ \ k \neq x \cdot n$

ggT(n, k) der größte gemeinsame Teiler von n und k

kgV(n, k) das kleinste gemeinsame Vielfache von n und k

n mod d Divisionsrest, wenn man n durch d teilt

ϕ(m) EULERsche **ϕ**-Funktion
(gibt die Anzahl derjenigen natürlichen Zahlen $n < m$ an, die teilerfremd zu m sind; $m, n \in \mathbf{N}$)

a und b teilerfremd heißt: **ggT**(a, b) = 1

Anmerkungen zum Sprachgebrauch:

- Eine **Algebra** (Plural: Algebren) definiert Elemente und Operatoren sowie Regeln bzw. Eigenschaften (**Axiome**) in bezug auf deren Verknüpfung (abgeschlossene Arithmetik).
- Eigenschaften sind beispielsweise die Existenz eines **neutralen Elementes e** (0 oder 1) oder die Existenz von **inversen Elementen a′** (– a bzw. a^{-1}).
- Regeln betreffen die **Assoziativität** und die **Kommutativität** bzgl. eines Operators oder die **Distributivität** bzgl. zweier Operatoren.
- Wir benutzen im folgenden die Operatoren + (**Addition**) und • (**Multiplikation**) sowie die Quantoren ∃ (**Existenzquantor**) und ∀ (**Allquantor**).

  Beispiele: $\exists\, x \in B : p(x)$ ist wahr! $\forall\, x \in B : p(x)$ ist wahr!

$< \mathbf{M}, \circ >$:= algebraisches System (hier: **Monoid**)

M = Nichtleere Menge {a, b, c, ...} der Operanden

$\circ$ = Operator (gebildet auf Elementpaare aus **M**)
hier: $\circ = \{+ , \bullet \}$

mit

1) $a, b \in \mathbf{M} \Rightarrow a \circ b \in \mathbf{M}$

2) $a, b, c \in \mathbf{M} \Rightarrow (a \circ b) \circ c = a \circ (b \circ c)$ **Assoziativ-Gesetz**

3) $a \in \mathbf{M} \Rightarrow \exists\, \mathbf{e}$ mit $a \circ \mathbf{e} = \mathbf{e} \circ a = a$

d. h. es existiert ein **neutrales Element e** (0 bzw. 1) in bezug auf die Operatoren + und $\bullet$.

$< \mathbf{G}, \circ >$:= algebraisches System (hier: **Gruppe**)

G = Nichtleere Menge {a, b, c, ...} der Operanden

$\circ$ = Operator (gebildet auf Elementpaare aus **G**)
hier: $\circ = \{+ , \bullet \}$

wenn

1) $< \mathbf{G}, \circ >$ ist ein **Monoid**, d. h. $\exists$ **e** = 0 bzw. 1

2) $a, b \in \mathbf{G} \Rightarrow a \circ b = b \circ a$ **Kommutativ-Gesetz**

(vgl. Abelsche Gruppe)

3) $a \in \mathbf{G} \Rightarrow \exists\ \mathbf{a'}$ mit $a \circ \mathbf{a'} = \mathbf{e}$

d. h. es existiert ein **inverses Element a′** = – a bezüglich der Addition und ein **inverses Element a′** = a^{-1} bezüglich der Multiplikation.

$< \mathbf{R}, +, \bullet >$:= algebraisches System (hier: **Ring**)

R = Nichtleere Menge {a, b, c, ...} der Operanden

$+, \bullet$ = Operatoren (gebildet auf Elementpaare aus **R**)

wenn

1) $< \mathbf{R}, + >$ ist eine **kommutative Gruppe**, d. h. $\exists\ \mathbf{e} = 0\ \ \&\ \ \exists\ \mathbf{a'} = -a$

2) $a, b, c \in \mathbf{R} \Rightarrow a \bullet (b + c) = (a \bullet b) + (a \bullet c)$ **Distributiv-Gesetz**

3) $< \mathbf{R}, \bullet >$ ist ein **Monoid**, d. h. für $a \in \mathbf{R}$: $\exists\ \mathbf{e} = 1$ aber $\overline{\exists}\ \mathbf{a'} = a^{-1}$

d. h. es existiert zwar ein **inverses Element a′** = – a bezüglich der Addition aber keine **multiplikative Inverse**.

$< \mathbf{K}, +, \bullet >$:= algebraisches System (hier: **Körper**)

$\mathbf{K}$ = Nichtleere Menge {a, b, c, ...} der Operanden

$+, \bullet$ = Operatoren (gebildet auf Elementpaare aus $\mathbf{K}$)

... ist ein **kommutativer Ring** mit Einselement, in dem jedes von Null verschiedene Element **invertierbar** ist, d. h. für

$$a \in \mathbf{K} \text{ mit } a \neq 0 \quad \Rightarrow \quad \exists\, \mathbf{a^{-1}} \in \mathbf{K} \text{ mit } a \bullet \mathbf{a^{-1}} = 1$$

Satz:

Jeder Körper $< \mathbf{K}, +, \bullet >$ und jeder Ring $< \mathbf{R}, +, \bullet >$ ist **nullteilerfrei** (Integritätsbereich), d. h. für $\forall\, x, y \in \mathbf{K} \setminus \{0\}$ bzw. $\forall\, x, y \in \mathbf{R} \setminus \{0\}$ gilt:

$$x \bullet y \neq 0$$

Satz:

$\mathbf{Z}_m := \{0, 1, ..., m - 1\}$ ist **ganau dann** ein Körper, wenn **m** eine Primzahl ist, d. h.: $\mathbf{Z}_p$ mit $p \in P$ ist ein Körper. (Beweis siehe Übung!)

Lemma:

Jeder Körper ist ein Ring. (Beweis siehe Übung!)

Definition:

Eine Algebra < **HG**, ○ > heißt **Halbgruppe**, wenn sie in Bezug auf die Operation ○ dem Assoziativgesetz genügt.

Sie wird **kommutative Halbgruppe** genannt, wenn die Operation ○ zusätzlich kommutativ ist.

Struktur					Bez.	Formel (Axiom)	A
Algebra (A, + , •)		Algebra (A, +)					N+1 Z Q
Körper	Ring	abelsche Gruppe	additive Gruppe	HG	assoz.	a + (b + c) = (a + b) + c	x x x
					∃ 0	0 + a = a	- x x
					∃ - a	a + (- a) = 0	- x x
					komm.	a + b = b + a	x x x
					distri.	a (b + c) = a b + a c	x x x
		Algebra (A_0 , •)					
		abelsche Gruppe	multipl. Gruppe	HG	assoz.	a (b c) = (a b) c	x x x
	Einselem.				∃ 1	1 a = a	x x x
					∃ a^{-1}	$a\ a^{-1} = 1$	- - x
					komm.	a b = b a	x x x

Restklassenring ist ein algebraisches System mit einer nichtleeren Menge von Elementen und **zwei** Operatoren ($\oplus$, $\otimes$), die auf die Elemente angewendet werden.

Anmerkung:

Wir sprechen oft nur von einem Ring $\mathbf{Z_m} := \{0, 1, 2, 3, \cdots, m - 1\}$, meinen aber den Restklassenring $< \mathbf{Z_m} , \oplus , \otimes >$, d. h. einen kommutativen Ring – auch „Ring von Integers modulo m“ genannt. Es ist heute bei weitem das wichtigste algebraische System der modernen Kryptographie.

Beispiel: $\mathbf{Z_2} := \{0, 1\}$; es gilt das Assoziativ-, Kommutativ- und Distributiv-Gesetz!

$0 \oplus 1 = 1$	$0 \otimes 1 = 0$
$1 \oplus 0 = 1$	$1 \otimes 0 = 0$
$0 \oplus 0 = 0$	$0 \otimes 0 = 0$
$1 \oplus 1 = 0$	$1 \otimes 1 = 1$

Ganzzahlige Division (Division mit Rest)

- Beim Dividieren einer natürlichen Zahl a durch eine natürliche Zahl b bleibt ein Rest $r \in \{0, 1, ..., b - 1\}$.

  Beispiel: $17 : 3 = 5$ Rest **2** weil $17 = 5 \cdot 3 + \mathbf{2}$

  $12 : 3 = 4$ Rest **0** weil $12 = 4 \cdot 3 + \mathbf{0}$

- Der Divisionsrest ist immer eindeutig und es gilt:

  Satz:

  Seien $a, b \in \mathbf{N}$ und $b > 0$. Dann gibt es eindeutige natürliche Zahlen q und r mit

  $$a = q \cdot b + r \text{ sowie } 0 \leq r < b.$$

  Man nennt q den Quotient, b den Divisor und r den Divisionsrest.

Mathematica

- In dem Computeralgebrasystem Mathematica gibt es die Funktionen

  Mod[a, b] und Quotient[a, b]

  mit der Eigenschaft

  $$a = \text{Quotient}[a, b] \cdot b + \text{Mod}[a, b].$$

- Quotient[a, b] $\in \mathbf{Z}$ ist der ganzzahlige Anteil von a / b. a und b dürfen beliebige reelle Zahlen sein mit $b \neq 0$.
- Zahlenbeispiel:

  Quotient[12345678, 3456] = 3572

  Mod[12345678, 3456] = 846

Definition: Seien $a, b \in \mathbf{Z}$ und sei $a = q \cdot b + r$ mit $0 \leq r < b$.

Dann schreibt man $r = a \bmod b$.

Zwei Zahlen $a, b \in \mathbf{Z}$ heißen **restgleich**, wenn $a \bmod n = b \bmod n$.

Man schreibt: $a \equiv b \bmod n$

und sagt: a ist **kongruent** zu b modulo n.

Beispiele:

$19 \equiv 12 \bmod 7 = 5$

2, 5, 8, 11, ... sind paarweise kongruent modulo 3

Definition: Sei $n \in \mathbf{N}$. Dann ist: $\mathbf{Z_n} := \mathbf{Z} / n\mathbf{Z} := \{0, 1, ..., n - 1\}$

Man spricht bei $\mathbf{Z_n}$ von einem **kommutativen Ring**.

Teilbarkeit

Es seien $n, d \in \mathbf{Z}$; $d \neq 0$.

- n heißt durch d teilbar $\Leftrightarrow$ $(\exists\, q \in \mathbf{Z})\ n = q \cdot d$
  Schreibweise: $\mathbf{d \mid n}$ (in Worten: d teilt n)
- Ist n nicht durch d teilbar, so schreiben wir $\mathbf{d \nmid n}$.
  $d = 0$ ist als Teiler nie zugelassen.

Satz:

$$0 \mid a \Leftrightarrow a = 0$$

$$a \mid b \text{ und } b \mid a \Rightarrow a = \pm b$$

$$(t \mid a \wedge t \mid b) \Rightarrow (\forall\, x, y \in \mathbf{Z})\ \ t \mid (a \cdot x + b \cdot y)$$

Euklidische Divisions-Theorem

Es seien $n, d \in \mathbf{Z}$; $d \neq 0$.

$\Rightarrow (\exists\, q, r \in \mathbf{Z})\ n = q \cdot d + r\ ;\ 0 \leq r < |d|$

q und r sind dabei eindeutig bestimmt.

Man nennt: d = Divisor; n = Divident; q = Quotient; r = Divisionsrest.

Notation: $r = R_d(n) = n \bmod d$

Sätze: $R_{-d}(n) = R_d(n)$

$\mathbf{R_d(n + i \cdot d) = R_d(n)}$ **Fundamental Property of Remainders!**

$(i \in \mathbf{Z})$

Größte gemeinsame Teiler

Definition: Der größte gemeinsame Teiler der Integerzahlen n_1 und n_2 (nicht beide gleich Null) ist der größte Integerwert t, der sowohl n_1 als auch n_2 teilt.

t heißt größter gemeinsamer Teiler von n_1 und n_2 .

$\Rightarrow \; (\exists\, t = \max\{\tau\}) \mid (\tau \mid n_1 \wedge \tau \mid n_2)$

Notation: **$t := ggT(n_1, n_2)$**

Sätze: $1 \le ggT(n_1, n_2) \le \min\{|n_1|, |n_2|\}$

$ggT(n_1, 0) = |n_1|$

Weitere Sätze:

Ist $ggT(a, b) = 1$, so werden a und b auch teilerfremd oder relativ prim bezeichnet.

$$ggT(n_1, n_2) = ggT(n_1 + i \cdot n_2, n_2)$$

Fundamental Property of Greatest Common Divisors!

$$ggT(n_1, n_2) = ggT(n_2, R_{n2}(n_1))$$

Euklid´s gcd Recursion!

Trickreich und nützlich sind noch folgende Sätze, bei denen $a, b, c \in Z$ gilt:

- $a \mid c$ und $b \mid c \Rightarrow a \cdot b \mid c \cdot ggT(a, b)$
- $a \mid b \Leftrightarrow ggT(a, b) = \mid a \mid$
- $ggT(a \cdot c, b \cdot c) = \mid c \mid \cdot ggT(a, b)$
- $a \mid b \cdot c$ und $ggT(a, b) = 1 \Rightarrow a \mid c$

(Zugehörige Beweise siehe Übungen!)

Kleinste gemeinsame Vielfache

Lösungsverfahren:

Das kleinste gemeinsame Vielfache von zwei ganzen Zahlen a und b können wir vereinfacht so bestimmen:

1. Wähle die größere der beiden Zahlen aus.
2. Bilde das 1fache, 2fache, 3fache, ... dieser Zahl. Prüfe jeweils, ob es auch ein Vielfaches der anderen Zahl ist.
3. Das erste gemeinsame Vielfache ist das sogenannte kleinste gemeinsame Vielfache **kgV** der beiden Zahlen.

Satz: $kgV(a, b) \cdot ggT(a, b) = a \cdot b$

Beispiel:

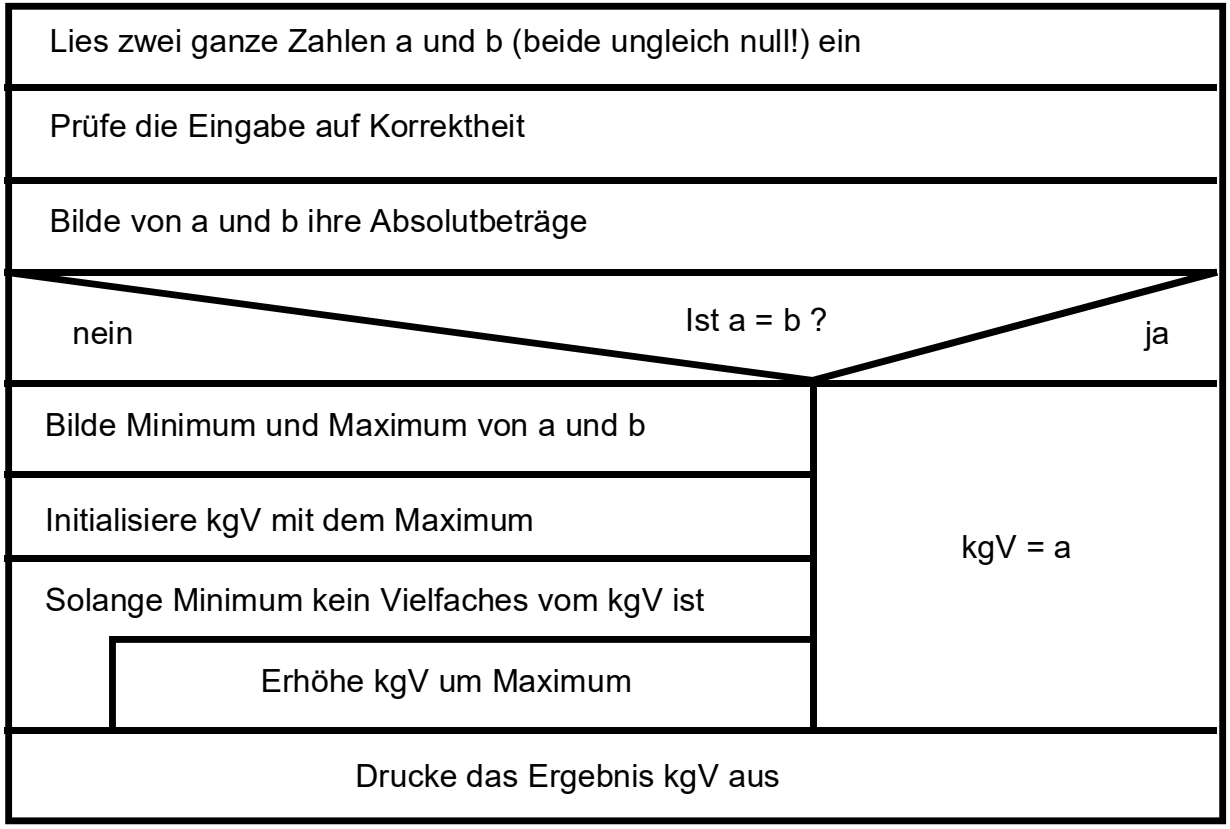
Berechnung von kgV(10, 8)

1. Wähle 10.
2. Prüfe:

   Ist 10 Vielfaches von 8? → nein
   Ist 20 Vielfaches von 8? → nein
   Ist 30 Vielfaches von 8? → nein
   Ist 40 Vielfaches von 8? → ja
3. Ergebnis: **kgV(10, 8) = 40**

Überprüfung: Weil ggT(10, 8) = 2 $\Rightarrow$ $kgV(a, b) \cdot ggT(a, b) = 80 = a \cdot b$

Struktogramm:



Kongruenzen

Es seien $n, r, d \in \mathbf{Z}$; $d \neq 0$.

Dann heißen n und r kongruent modulo d $\Leftrightarrow d \mid (n - r)$

$\Leftrightarrow (\exists\, x \in \mathbf{Z})\ n = x \cdot d + r$

Schreibweise: $\mathbf{n \equiv r \pmod{d}}$

Ist n – r nicht durch d teilbar, so heißen n und r inkongruent modulo d.

Schreibweise: $\mathbf{n \not\equiv r \pmod{d}}$

Sätze: $a \equiv b \pmod{m} \Leftrightarrow b \equiv a \pmod{m} \Leftrightarrow a - b \equiv 0 \pmod{m}$

$a \equiv b \pmod{m} \Rightarrow ggT(a, m) = ggT(b, m)$

Restsystem

Definition: y heißt ein Rest von x modulo m $\Leftrightarrow x \equiv y \pmod m$.
Eine Menge von Zahlen $x_1, x_2, \ldots, x_m$ heißt vollständiges Restsystem modulo m, falls für alle $y \in \mathbf{Z}$ genau ein x_j ($j \in [1:m]$) mit

$$\mathbf{y \equiv x_j \pmod m}$$

existiert.

Satz: Es gibt unendlich viele vollständige Restsysteme modulo m.
$\mathbf{Z_m} := \{0, 1, 2, \ldots, m - 1\}$ bilden ein vollständiges Restsystem.

Sätze von Fermat und Euler

Es sei $p \in \mathbf{P}$ und $a \in \mathbf{Z}$.

$$ggT(a, p) = 1 \quad \Rightarrow \quad a^{p-1} \equiv 1 \pmod{p}$$

Für $\forall\, b \in \mathbf{Z}$ gilt: $b^{p} \equiv b \pmod{p}$

$$\phi(p) = p - 1$$

$$\mathbf{\phi(p \cdot q) = (p - 1)(q - 1)}$$

für $p \neq q$ und $p, q \in \mathbf{P}$

Eulersche Verallgemeinerung:

Es sei $a \in \mathbf{Z} \setminus \{0\}$ und $m \in \mathbf{N} + \mathbf{1}$.

$$ggT(a, m) = 1 \quad \Rightarrow \quad a^{\phi(m)} \equiv 1 \pmod{m}$$

Quadratische Reste

Es sei a, m $\in$ **N + 1** mit ggT(a, m) = 1 .

a heißt quadratischer Rest modulo m, falls:

$$\mathbf{x^2 \equiv a \pmod{m}}$$ lösbar ist.

a heißt quadratischer Nichtrest modulo m, falls:

$$\mathbf{x^2 \equiv a \pmod{m}}$$ nicht lösbar ist.

Satz: Es sei p $\in$ **P**; a$\in$ **Z** mit ggT(a, p) = 1 und

$$\mathbf{x^2 \equiv a \pmod{p}} \qquad (*)$$

Wenn $a^{(p-1)/2} \equiv 1 \pmod{p}$ $\Leftrightarrow$ (*) hat 2 Lösungen x_1 und x_2.

$a^{(p-1)/2} \equiv -1 \pmod{p}$ $\Leftrightarrow$ (*) hat keine Lösung.

Der diskrete Logarithmus

Gegeben seien $g, \alpha \in \mathbf{Z}$ und $p \in \mathbf{P}$.

Gesucht ist x, so daß

$$g^x = \alpha \bmod p$$

- Das Problem ist äußerst schwierig zu lösen, falls p mindestens 100 Dezimalstellen hat.
- Der Sicherheitswert vieler Algorithmen beruht auf der Schwierigkeit des diskreten Logarithmus.

Anwendungen des diskreten Logarithmus

- Schlüsselaustausch (Diffie-Hellmann)
- Verschlüsselung (El Gamal, Massey-Omura) und
- Digitale Signatur (El Gamal, DSS)

Nicht nur diskreter Logarithmus „modulo p“, sondern auch

- In beliebigen endlichen Körpern
  (mit p oder 2^n Elementen)
- Auf „elliptischen Kurven“
- Allgemein in „Gruppen“

Satz 1:

Die „Diskrete Exponentialfunktion“ ist eine Einwegfunktion.

Satz 2:

Mit den besten bekannten Algorithmen braucht man i. a. exponentiellen Aufwand, um aus **y** ein **x** mit

$$\mathbf{y = g^x \bmod p}$$

zu berechnen.

Beispiel:

g = 2
p = 37

x	5	10	20	30
y	32	25	33	11

Wir suchen eine Lösung x der **quadratischen Gleichung**

$$x^2 \equiv a \pmod{p} \qquad (1)$$

in der Ringstruktur $\mathbf{Z_p}$ mit $p \in \mathbf{P}$.

Eine solche Lösung bezeichnen wir als **modulare Quadratwurzel** und schreiben sie als:

$$x \equiv \sqrt{a} \pmod{p} \qquad (2)$$

Zahlen $a \in \mathbf{Z_p}$, welche eine modulare Quadratwurzel besitzen, nennen wir quadratische Reste (sonst quadratische Nichtreste).

Nach heutigem Kenntnisstand ist die Berechnung der modulare Quadratwurzel (mod p) ein **recht schwieriges** mathematisches Problem.

Satz:

Modulare Quadratwurzeln (mod p) treten immer **paarweise** auf.
Beweis hierzu siehe im folgenden.

Satz:

Damit Gl(1) eine Lösung x $\in$ $\mathbf{Z_p}$ besitzt, muss gelten:

$$p \equiv 3 \pmod 4 \qquad (3)$$

oder

$$a^{\frac{p-1}{2}} \bmod p = +1 \qquad (4)$$

Ansonsten existiert keine Lösung.

Satz:

Unter der Voraussetzung von Gl(3) bzw. Gl(4) ergibt sich für Gl(1) als eine Quadratwurzel die Lösung:

$$x_1 = a^{\frac{p+1}{4}} \bmod p \qquad (5)$$

Eine weitere Lösung (zweite Quadratwurzel) von Gl(1) ist dann auch gegeben durch:

$$x_2 = p - x_1 \qquad (6)$$

Vorstehende Sätze sollen im folgenden bewiesen werden. Dazu benötigen wir lediglich den Satz von Fermat.

Beweis für x_1:

Aus dem Satz von Fermat

$$a^{p-1} \bmod p = 1$$

folgt nach Multiplikation mit a^2:

$$a^{p+1} \bmod p = a^2 \quad \Big| \quad ()^{1/4}$$

$$\Rightarrow a^{\frac{p+1}{4}} \bmod p = a^{\frac{1}{2}} = \sqrt{a}$$

bzw.

$$x_1 := \sqrt{a} = a^{\frac{p+1}{4}} \bmod p \qquad q.e.d.$$

Beweis für x_2:

Aus Gl(1) ergibt sich:

$$x \equiv \pm\sqrt{a} \pmod{p}$$

Ist $x = x_1$ eine Lösung von Gl(1), d. h. $x_1 = +\sqrt{a} \bmod p$, dann ist auch $x_2 = -\sqrt{a} \bmod p = -x_1$ eine Lösung.

Und somit:

$$x_2 \equiv -x_1 \bmod p = (p - x_1) \bmod p \qquad q.e.d.$$

Beweis für die ∃ von $x_{1,2}$:

Eine Lösung nach Gl(5) ist nur **berechenbar**, falls der Exponent (p + 1)/4 eine ganze Zahl ist.

$$\Rightarrow \quad p+1 \ \textit{ist ein Vielfaches von } 4$$

$$\Rightarrow \quad 4 \ \Big| \ (p+1)$$

$$\Rightarrow \quad p+1 = 4 \cdot k \quad (k = 1, 2, \dots)$$

$$\Rightarrow \quad p = 4 \cdot k - 1$$

$$\Rightarrow \quad p = \{3, 7, 11, 19, 23, 31, \dots\}$$

$$\Rightarrow \quad p \equiv 3 \ (\text{mod } 4) \qquad \textit{q.e.d.}$$

Wir suchen eine Lösung x für die modulare Quadratwurzel

$$x^2 \equiv a \pmod{n} \qquad (7)$$

in der Ringstruktur $\mathbf{Z_n}$ mit $\mathbf{n = p \cdot q}$ und p, q ∈ **P**.

Satz:

Genügen p und q den Bedingungen

$$p \equiv 3 \pmod 4 \quad bzw. \quad q \equiv 3 \pmod 4,$$

so besitzt die Gl(7) dann insgesamt **vier** Quadratwurzeln der Form **± r** und **± s** in der Menge {0, 1, 2, …, n - 1}, also mod n.

Beweis siehe Übungsaufgaben!

Kap. 2: Algebraische Strukturen und elementare Zahlentheorie

Teil 2: Zahlentheorie und kryptologische Anwendungen

- Zahlentheoretische Funktionen und Algorithmen
  - Euklidischer und erweiterter Euklidischer Algorithmus
  - Sätze von Fermat und Euler
  - Modulare Inverse und modulare Exponentiation
- Generierung von Schlüsselparametern
  - Sieb von Eratosthenes
  - Rückgekoppelte Schieberegister
  - Blum-Micali-Generator
- Bitstromverschlüsselung

Definition: Für a, b $\in$ **N** sei g = ggT(a, b) der größte gemeinsame Teiler von a und b, d. h. die größte ganze Zahl, die a und b ohne Rest teilt.

Pseudocode:

```
repeat
    r := a mod b
    if r == 0 then
        g := b
    else
    {   a = b
        b = r   }
until r == 0
```

Satz: Zwei natürliche Zahlen a und b heißen **relativ prim** oder **teilerfremd**, wenn ggT(a, b) = 1 ist.

Berechnungsbeispiel:

Der ggT von 531 und 93 lässt sich wie folgt durch wiederholte Division berechnen:

$$
\begin{aligned}
531 &= 5 \cdot 93 + 66 \\
93 &= 1 \cdot 66 + 27 \\
66 &= 2 \cdot 27 + 12 \\
27 &= 2 \cdot 12 + 3 \\
12 &= 4 \cdot 3
\end{aligned}
$$

$\Rightarrow$ $\underline{\underline{ggt(531, 93) = 3}}$

Kurz:

a	b	g
531	93	66
93	66	27
66	27	12
27	12	3
12	3	0

Satz (Lineare diophantische Gleichung):

Sei a, b, d $\in$ **Z** und d = ggT(a, b) > 0. Dann gibt es ganze Zahlen **x** und **y** mit:

$$d = a \cdot \mathbf{x} + b \cdot \mathbf{y} ,$$

wobei: $|\mathbf{x}| \leq b / (2 \cdot d)$ und $|\mathbf{y}| \leq a / (2 \cdot d)$.

Pseudocode (Berlekamp-Algorithmus):

```
ErwEuklid(a, b)
if b == 0 then return (a, 1, 0)
(d, x, y) = ErwEuklid(b, Mod(a, b))
return (d, y, x - Div(a, b) * y)
```

Hilfsfunktionen:

```
Mod(a, n)              Div(a, n)
return (a % n)         return ((a - Mod(a, n)) / n)
```

Berechnungsbeispiel:

Gesucht sei die Lösung der Gleichung

$$3 = 531 \cdot x + 93 \cdot y \quad (*)$$

Durch schrittweises Rückwärts-einsetzen ergibt sich aus

$$531 = 5 \cdot 93 + 66 \quad (1)$$
$$93 = 1 \cdot 66 + 27 \quad (2)$$
$$66 = 2 \cdot 27 + 12 \quad (3)$$
$$27 = 2 \cdot 12 + 3 \quad (4)$$

folgende Gleichungskette:

$$3 \overset{(4)}{=} 27 - 2 \cdot 12$$
$$\overset{(3)}{=} 27 - 2 \cdot (66 - 2 \cdot 27) = -2 \cdot 66 + 5 \cdot 27$$
$$\overset{(2)}{=} -2 \cdot 66 + 5 \cdot (93 - 1 \cdot 66) = 5 \cdot 93 - 7 \cdot 66$$
$$\overset{(1)}{=} 5 \cdot 93 - 7 \cdot (531 - 5 \cdot 93) = -7 \cdot 531 + 40 \cdot 93$$

$\Rightarrow$ Zahl 3 darstellbar als Linear-kombination von 531 und 93.

Durch Vergleich mit (*) erhält man

$$x = -7 \quad \text{und} \quad y = 40$$

Modulare Inversion:

Wir nehmen an, dass ggT(a, n) = 1 ist. Dann gibt es ganze Zahlen x und y mit $1 = a \cdot x + n \cdot y$ und es folgt:

$$x = a^{-1} \bmod n \ .$$

Mit dem erweiterten Euklidischen Algorithmus (Berlekamp-Algorithmus) haben wir also ein Verfahren zur Berechnung der Inversen $a^{-1} \bmod n$.

Beispiel:

Gesucht ist die Inverse von 11 in $\mathbf{Z}_{26}$.

$\Rightarrow 11^{-1} \bmod 26 = 19$ weil $19 \cdot 11 \bmod 26 = 1$

Herleitung:

$ggT(a,b) = a \cdot x + b \cdot y \Rightarrow x, y$

1. Setze: $b = n$
2. Annahme: $ggt(a,n) = 1$

$\Rightarrow \quad 1 = a \cdot x + n \cdot y$

3. Bilde: mod n auf beiden Seiten

$$\underbrace{1 \bmod n}_{=1} = a \cdot x \bmod n + \underbrace{n \cdot y \bmod n}_{=0}$$

Also $1 = a \cdot x \bmod n$

$\Leftrightarrow \quad x = a^{-1} \bmod n$

Berechnungsbeispiel:

gesucht : Inverse von 11 in $\mathbf{Z}_{26}$ = ?

(falls sie existiert)

a	b	g
11	26	11
26	11	4
11	4	3
4	3	1

a = 11 b = n = 26

$26 = 2 \cdot 11 + 4 \quad (1)$

$11 = 2 \cdot 4 + 3 \quad (2)$

$4 = 1 \cdot 3 + 1 \quad (3)$

Da ggt(11,26) = 1 ist 11 invertierbar.

Nun wird schrittweise rückwärts eingesetzt:

$1 \overset{(3)}{=} 4 - 1 \cdot 3$

$\overset{(2)}{=} 4 - 1 \cdot (11 - 2 \cdot 4) = -11 + 3 \cdot 4$

$\overset{(1)}{=} -11 + 3 \cdot (26 - 2 \cdot 11)$

$= \underbrace{3}_{y} \cdot 26 - \underbrace{7}_{x} \cdot 11$

Also ist

$a^{-1} \bmod n = 11^{-1} \bmod 26$

$= -7 \bmod 26$

$= 19$

Probe: $11 \cdot 19 \bmod 26$

$= 209 \bmod 26 = 1$

Definition: Für jede natürliche Zahl n gibt die Eulersche ϕ-Funktion ϕ(n) die Anzahl der natürlichen Zahlen kleiner als n an, die zu n **teilerfremd** sind, d. h.

$$\phi(n) = \left| \; 0 \le k < n \mid ggT(k, n) = 1 \; \right|$$

Beispiel:

ϕ(15) = 8 , denn die Zahlen 1, 2, 4, 7, 8, 11, 13 und 14 sind teilerfremd zu 15.

Spezialfälle:

- $p \in \mathbf{P} \quad \Rightarrow \quad \phi(p) = p - 1$
- $k \in \mathbf{N+1} \quad \Rightarrow \quad \phi(p^k) = p^{k-1} \cdot (p - 1)$
- $p, q \in \mathbf{P} \quad \Rightarrow \quad \phi(p \cdot q) = (p - 1) \cdot (q - 1)$
  $(p \neq q) \qquad\qquad\qquad\quad = \phi(p) \cdot \phi(q)$

Berechnungsbeispiel:

gesucht : Φ(15) = ?

Berechne :

		# ↓ abzählen :
ggT(0,15)	≠ 1	
ggT(1,15)	= 1	1
ggT(2,15)	= 1	2
ggT(3,15)	≠ 1	
ggT(4,15)	= 1	3
ggT(5,15)	≠ 1	
ggT(6,15)	≠ 1	
ggT(7,15)	= 1	4
ggT(8,15)	= 1	5
ggT(9,15)	≠ 1	
ggT(10,15)	≠ 1	
ggT(11,15)	= 1	6
ggT(12,15)	≠ 1	
ggT(13,15)	= 1	7
ggT(14,15)	= 1	8

ggt = 1 ist 8

⇒ Φ(15) = 8

Spezialfall : Φ(p) = p−1

p ∈ P

Satz: Wenn ggT(a, n) = 1 ist, dann gilt:

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

Ist $n = p \in \mathbf{P}$ eine Primzahl, so ergibt sich der Fermatsche Satz:

$$a^{p-1} \bmod p = 1 \; ; (a \neq 0)$$

und damit auch für die modulare Inverse:

$$a^{-1} \bmod p = a^{p-2} \bmod p \; ; (a \neq 0)$$

Beispiel: $6^{-1} \bmod 23 = 6^{21} \bmod 23 = \underline{\underline{4}}$

Berechnungsbeispiel:

gesucht : $6^{-1} = ?$ (mod 23)

Satz von Fermat:

$a^{p-1} \bmod p = 1$; ($a \neq 0$, $p \in P$)

$a \cdot a^{-1} \cdot a^{p-1} \bmod p = 1$

($a \cdot a^{-1} = 1$, a^{p-2})

also

$a \cdot a^{p-2} \bmod p = 1$

Auf der anderen Seite gilt :

$a \cdot a^{-1} \bmod p = 1$

also

$$a^{-1} \bmod p = a^{p-2} \bmod p$$

d.h.

$$6^{-1} \bmod 23 = \underbrace{6^{21} \bmod 23}_{=4}$$

$$6^{-1} \bmod 23 = 4$$

Probe: $6 \cdot 4 = 24 \equiv 1$ (mod 23)

q.e.d.

Satz: Wenn ggT(a, n) = 1 ist, dann gilt:

$$a^b \bmod n = a^{b \bmod \phi(n)} \bmod n$$

Beweis:

Entwickle b als Quotient von $\phi(n)$, d. h.

$$b = q \cdot \phi(n) + r \quad \text{mit} \quad r = b \bmod \phi(n) < \phi(n).$$

Also

$$\begin{aligned} a^b \bmod n &= a^{q \cdot \phi(n)} \bmod n \cdot a^r \bmod n \\ &= \underbrace{a^{\phi(n)} \bmod n \cdot a^{\phi(n)} \bmod n \cdot \ldots \cdot a^{\phi(n)} \bmod n}_{q \text{ mal}} \cdot a^r \bmod n \\ &= 1 \cdot 1 \cdot \ldots \cdot 1 \cdot a^r \bmod n \\ &= a^r \bmod n = a^{b \bmod \phi(n)} \bmod n \qquad \text{q.e.d.} \end{aligned}$$

Anwendungsbeispiel:

$a^{1234} \bmod 31 = a^{4} \bmod 31$

Nebenrechnung:

$1234 \bmod \Phi(31)$

$= 1234 \bmod 30 = 4$

Berechnungsbeispiel:

$a^b \bmod n = a^{b \bmod \Phi(n)} \bmod n$

a = 1234 n = p = 163

b = 5678

n = 163

$1234^{5678} \bmod 163 = ?$

$\Phi(n) = \Phi(163) = 162$

$b \bmod \Phi(n) = 5678 \bmod 162 = \mathbf{8}$

$\rightarrow 1234^{5678} \bmod 163 =$

$1234^{8} \bmod 163 =$

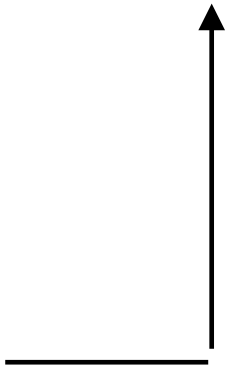
$((1234^2)^2)^2 \bmod 163 = 57$

$1234 \bmod 163 = 93$

$93^2 \bmod 163 = 10$

$10^2 \bmod 163 = 100$

$100^2 \bmod 163 = 57$



Beispiel: Wir betrachten ($\mathbf{Z}_4$, +, •) und erstellen die Verknüpfungstabelle für die Multiplikation:

•	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	**0**	2
3	0	3	2	1

- $\mathbf{Z}_4$ ist nicht nullteilerfrei und es gibt zu 2 kein multiplikatives Inverses.
- Die Zahl 3 ist zu sich selbst invers, denn $3 \cdot 3 = 1$.

Beispiel: Wir betrachten ($\mathbf{Z}_3$, +, •) und erstellen ebenfalls die Verknüpfungstabelle für die Multiplikation:

•	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

- Die Zahl 2 ist invers zu sich selbst, denn $2 \cdot 2 = 1$, d. h. $2^{-1} = 2$.
- Wegen $2 \cdot ½ = 1$ folgt in $\mathbf{Z}_3$ auch: $½ = 2$
  (mit ½ ist nicht die Zahl 0.5 gemeint, die es in $\mathbf{Z}_3$ nicht gibt!)

Zur effizienten Berechnung von $a^b \bmod n$ wird bei binärer Repräsentation des (k+1)-Bit-Exponenten $b = (b_k, ..., b_1, b_0)$ die Funktion ModExpo(a, b, n) benutzt.

Pseudocode (Square-and-Multiply-Algorithmus):

```
d := 1
   for i := k to 0 do
   {   d := (d * d) mod n
       if bi == 1 then
          d := (d * a) mod n   }
return d
```

Ausgabe: $d = a^b \bmod n$

(maximal *size(Exponent)* Quadrierungen und Multiplikationen!)

Berechnungsbeispiel:

gesucht : $6^{21} \bmod 23 = ?$

$\Rightarrow$ a = 6 und b = 21 = 16 + 4 + 1

$= (1\ 0\ 1\ 0\ 1)_d$

$b_4\ \ b_2\ \ b_0$

$\Rightarrow$ 8 Multiplikationen

d · d oder d · a

d = 1

for i = 4 to 0

d = d · d mod n = 1

if $b_4 == 1$ then d:= d·a = a (weil $b_4 = 1$)

d = d · d mod n = $a^2 \bmod n$

if $b_3 == 1 \Rightarrow$ nein

d = d · d mod n = $a^2 \cdot a^2 \bmod n = a^4 \bmod n$

if $b_2 == 1$ then d:= d·a = a^5 (weil $b_2 = 1$)

d = d · d mod n = $a^5 \cdot a^5 \bmod n = a^{10}$

if $b_1 == 1 \Rightarrow$ nein

d = d · d mod n = $a^{10} \cdot a^{10} \bmod n = a^{20}$

if $b_0 == 1$ then d:= d·a = a^{21} (weil $b_0 = 1$)

Berechnungsbeispiel:

gesucht : $6^{21} \bmod 23 = ?$

$\Rightarrow$ a = 6 und b = 21 = 16 + 4 + 1

$= (1\ 0\ 1\ 0\ 1)_d$

$b_4\ \ b_2\ \ b_0$

$\Rightarrow \quad \underline{\underline{6^{21} \equiv 4 \bmod 23}}$

(erzielt nach 8 Multiplikationen)

d = 1

for i = 4 to 0

d = 1 · 1 mod 23 = 1

if b_4 == 1 then d:= 1·6 = 6 (weil b_4 = 1)

d = 6 · 6 mod 23 = 6^2 mod 23 = 13

if b_3 == 1 $\Rightarrow$ nein

d = 13 · 13 mod n = 13^2 mod 23 = 8

if b_2 == 1 then d:= 8·6 = 2 (weil b_2 = 1)

d = 2 · 2 mod 23 = 4

if b_1 == 1 $\Rightarrow$ nein

d = 4 · 4 mod 23 = 16

if b_0 == 1 then d:= 16·6 = 4 (weil b_0 = 1)

Problemstellung:

$x \equiv a_i \bmod m_i$; $i = 1, 2, ...$

Beispiel :

$x \bmod 4 = 2$

$x \bmod 3 = 1$

$x \bmod 5 = 0$

gesucht : $x = ?$

(kleinste natürliche Zahl)

Antwort / Lösung : $x = 10$

Fragestellungen :

1. Wie findet man x ?
2. Lässt sich x eindeutig benennen ?
3. Effiziente Berechnung möglich ?

$\Rightarrow$ Lösungsalgorithmus

Lösungsmethode:

Seien $m_1, m_2, ..., m_n \in \mathbf{N+1}$ und paarweise teilerfremd
sowie $a_1, a_2, ..., a_n \in \mathbf{Z}$.

Simultane Kongruenz:

$$x \equiv a_1 \bmod m_1 \; ; \; x \equiv a_2 \bmod m_2 \; ; \; \ldots \; ; \; x \equiv a_n \bmod m_n \qquad (1)$$

Lösung:

1. Setze: $m = \prod_{i=1}^{n} m_i$; $M_i = m / m_i$ $(1 \le i \le n)$
   d. h., in M_i ist kein m_i enthalten!
2. Weil die m_i paarweise teilerfremd sind, gilt:
   $ggT(m_i, M_i) = 1$ für $1 \le i \le n$.

3. Nutze den erweiterten Euklidischen Algorithmus, um Zahlen $y_i \in \mathbf{Z}$ für $1 \le i \le n$ zu berechnen:

$$y_i \cdot M_i \equiv 1 \pmod{m_i} \Leftrightarrow y_i \cdot M_i + n_i \cdot m_i = 1 \quad \text{für} \quad (1 \le i \le n) \qquad (2)$$

Behauptung: Eindeutige Lösung der simultanen Kongruenz (1) ist:

$$x = \left(\sum_{i=1}^{n} a_i \cdot y_i \cdot M_i\right) \bmod m \qquad (3)$$

Beweis: Aus (2) folgt:

$$a_i \cdot y_i \cdot M_i \equiv a_i \pmod{m_i} \quad \text{für} \quad (1 \le i \le n) \qquad (4)$$

und weil für $j \ne i$ die Zahl m_i ein Teiler von M_j ist, gilt:

$$(a_j \cdot y_j \cdot m_1 \cdot m_2 \cdot m_i \cdot \cancel{m_j} \cdot m_n \,/\, \cancel{m_j}) \bmod m_i = 0 \quad \Leftrightarrow$$

$$a_j \cdot y_j \cdot M_j \equiv 0 \pmod{m_i} \quad \text{für} \quad (1 \le \{i, j\} \le n \quad \text{sowie} \quad i \ne j) \tag{5}$$

Aus dem Lösungsansatz gemäß (3) folgt:

$$x = \left(\sum_{i=1}^{n} a_i \cdot y_i \cdot M_i\right) \bmod m$$

$$= (a_i \cdot y_i \cdot M_i) \bmod m + \left(\sum_{\substack{j=1 \\ j \ne i}}^{n} a_j \cdot y_j \cdot M_j\right) \bmod m \,,$$

wobei wir die Summe in zwei Terme geschrieben haben!

Nun wenden wir auf der rechten und linken Seite die Operation **mod m_i** an:

$$x \bmod m_i = \left(\left(\sum_{i=1}^{n} a_i \cdot y_i \cdot M_i \right) \bmod m \right) \bmod m_i$$

$$= \left(\sum_{i=1}^{n} a_i \cdot y_i \cdot M_i \right) \bmod m_i$$

$$= (a_i \cdot y_i \cdot M_i) \bmod m_i + \left(\sum_{\substack{j=1 \\ j \neq i}}^{n} a_j \cdot y_j \cdot M_j \right) \bmod m_i$$

$\downarrow$ $\qquad$ $\downarrow$

$= a_i$ wegen (4) $\qquad$ $= 0$ wegen (5)

$\rightarrow$ $x \equiv a_i \pmod{m_i}$ für $(1 \le i \le n)$ **q.e.d.**

Also löst x nach (3) die Kongruenz (1)!

Rechenbeispiel:

$\mathbf{x} \equiv 2 \bmod 4$; $\mathbf{x} \equiv 1 \bmod 3$ und $\mathbf{x} \equiv 0 \bmod 5$; gesucht: $\mathbf{x} = ?$

$a_1 = 2$ $m_1 = 4$

$\leftrightarrow$ $a_2 = 1$ und $m_2 = 3$ $\rightarrow$ $m = m_1 \cdot m_2 \cdot m_3 = 60$ $\rightarrow$

$a_3 = 0$ $m_3 = 5$

$M_1 = 60 / 4 = 15;$

$M_2 = 60 / 3 = 20$ und

$M_3 = 60 / 5 = 12.$

Wir lösen:

1) $y_1 \cdot M_1 \equiv 1 \bmod m_1 \rightarrow y_1 \cdot 15 \equiv 1 \bmod 4 \rightarrow \mathbf{y_1 = 3}$

2) $y_2 \cdot M_2 \equiv 1 \bmod m_2 \rightarrow y_2 \cdot 20 \equiv 1 \bmod 3 \rightarrow \mathbf{y_2 = 2}$

3) $y_3 \cdot M_3 \equiv 1 \bmod m_3 \rightarrow y_3 \cdot 12 \equiv 1 \bmod 5 \rightarrow \mathbf{y_3 = 3}$

Mit (3) erhalten wir die Lösung:

$$\underline{\underline{\mathbf{x}}} = \left(\sum_{i=1}^{3} a_i \cdot y_i \cdot M_i\right) \bmod m$$

$$= (2 \cdot \mathbf{3} \cdot 15 + 1 \cdot \mathbf{2} \cdot 20 + 0 \cdot \mathbf{3} \cdot 12) \bmod 60 = \underline{\underline{\mathbf{10}}}$$

Algorithmus:

Pseudocode (Chinesischer Restalgorithmus):

```
CRT(int koeff[], int moduli[], int number)
{   int multi[number], result = 0, modulus, i
    CRTPre(moduli[], number, modulus, multi[])
    for i := 1 to number do
        result = (result + koeff[i] * multi[i]) % modulus;
return result
}
```

Benutzt wird die Hilfsfunktion `CRTPre`, die mittels `ErwEuklid` die Werte y_i und anschließend die Produkte $y_i \cdot M_i$ berechnet und in der Variable $multi_i$ übergibt.

Pseudocode (Hilfsfunktion):

```
CRTPre(moduli[], number, modulus, multi[])
{   int i, m, M, x, y, ggT, modulus = 1
    for i := 1 to number do
        modulus = modulus * moduli[i]
    for i := 1 to number do
        {   m = moduli[i]
            M = modulus / m
            (ggT, x, y) = ErwEuklid(M, m)
            multi[i] = (x * M) % modulus   }
return (modulus, multi[])   }
```

Für die Implementierung des erweiterten Euklidischen Algorithmus `ErwEuklid` (Berlekamp-Algorithmus) siehe Kap. IV Seite 38.

Seien $m_1, m_2, ..., m_n \in \mathbf{N+1}$ und paarweise teilerfremd sowie $a_1, a_2, ..., a_n \in \mathbf{Z}$.

Behauptung:

Dann hat die simultane Kongruenz gemäß (1)

$$x \equiv a_i \bmod m_i \quad (1 \le i \le n)$$

eine Lösung gemäß (3), die eindeutig ist mod m mit: $m = \prod_{i=1}^{n} m_i$.

Satz:

Der Algorithmus zur Lösung der simultanen Kongruenz erfordert mit k := size(m) folgende Aufwände:

Zeit-Aufwand:	$o(k^2)$
Speicherplatz-Aufwand:	$o(k)$

Schlüsselparameter

- Die **Sicherheit** aller kryptographischer Verfahren basiert auf der Schwierigkeit, einen geheimen Schlüssel zu erraten oder anderweitig zu beschaffen.
- Im Zusammenhang mit kryptographischen Schlüsselparameter spielen das Erzeugen von **Zufallszahlen** (möglichst zufällig) sowie die Ermittlung von **Primzahlen** (möglichst groß) eine zentrale Rolle.
- Ein **Pseudozufallszahlengenerator** ist ein Algorithmus, der nach Eingabe von Initialisierungsdaten (sogenannte seed numbers) eine Zufallsfolge deterministisch erzeugt.
- Zufallszahlen, die aus einem physikalischen Zufallsprozess (z. B. thermisches Rauschen oder radioaktiver Zerfall) abgeleitet werden, nennt man **echte Zufallszahlen**.

Formel:

$$x_{n+1} = (a \cdot x_n + b) \bmod n$$

x_0 = Startwert (seed number) $\in$ **N+1**
a, b, n = Parameter (konstant) $\in$ **N+1**
x_{n+1} = Pseudozufallszahlen (n = 0, 1, 2, ...)

Parameterwahl:

Parameter	Möglichkeit 1	Möglichkeit 2
a	137153	7141
b	17	54773
n	2^{19}	259200

Ausgabe:

$x_0, x_1, x_2, \ldots$

Pseudozufallszahlengenerator

Formel:

$$x_{n+1} = a \cdot x_n - \text{int}\,(a \cdot x_n / b) \cdot b$$

x_0 = Startwert (seed number) $\in$ **N+1**
a, b = Parameter (konstant) $\in$ **N+1**
int (...) = ganzzahliger Anteil Klammerausdruck
x_{n+1} = Pseudozufallszahlen (n = 0, 1, 2, ...)

Parameterwahl:

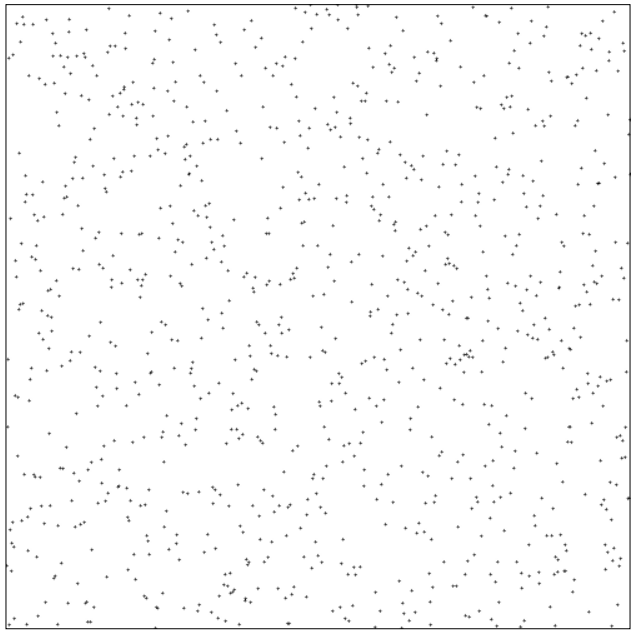
Parameter	Möglichkeit 1	Möglichkeit 2
a	23	29
b	100 000 001	1 000 001
x_0	439 147	691 156

Ausgabe:

$x_1, x_2, x_3, \ldots$

Integerzahlengenerator

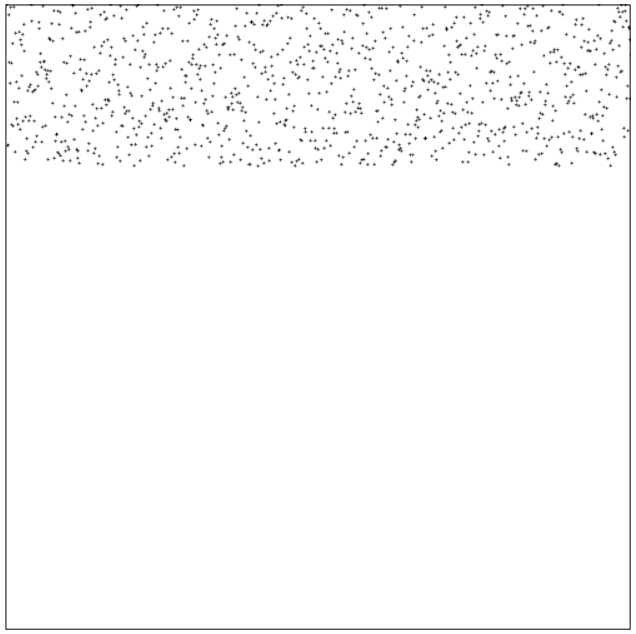
x_0 = 439147 a = 23 b = 100000001
(gerechnet von Marcell Dietl)



x-Achse: 1, 2, 3, …, 1000
y-Achse: x_1, x_2, x_3, …, x_{1000}

Linearer Kongruenzgenerator

x_0 = 4711 a = 7141 b = 54773
n = 259200



x-Achse: 1, 2, 3, …, 1000
y-Achse: x_1, x_2, x_3, …, x_{1000}

Definition: Es sei $p \in \mathbf{Z}$ und $p > 1$. p heißt **Primzahl** (auch unzerlegbar oder irreduziebel) $\Leftrightarrow$ Es gibt keine Teiler a von p mit $1 < a < p$.

Eine Zahl $a \in \mathbf{Z}$ mit $|a| \geq 2$ heißt **zerlegbar**, wenn $|a|$ keine Primzahl ist.

Satz:

1. Es sei $p \in \mathbf{Z}$ und $p > 1$. p ist genau dann eine Primzahl, wenn gilt:

$$\text{Aus } p \mid (a \cdot b) \Rightarrow p \mid a \text{ oder } p \mid b .$$

2. $\forall\, a \in \mathbf{N}$ mit $a \geq 2 \Rightarrow \exists$ Primzahl p mit $p \mid a$.
3. Jede Zahl $a \in \mathbf{N}$ mit $a \geq 2 \Rightarrow$ a ist das Produkt endlich vieler Primzahlen, d. h. es gilt die **Primfaktorzerlegung**:

$$a = p_1 \cdot p_2 \cdot \ldots \cdot p_k$$

Fundamentalsatz der Arithmetik

Jede natürliche Zahl $a > 1$ besitzt eine eindeutige Primfaktorzerlegung der Form:

$$a = p_1^{a_1} \cdot p_2^{a_2} \cdot \ldots \cdot p_k^{a_k}$$

wobei $p_1, \ldots, p_k \in \mathbf{P}$ und $a_1, \ldots, a_k \in \mathbf{N}$.

Satz:

Sei n eine **un**gerade Zahl für die auch $(n - 1) / 2$ **un**gerade ist. Dann gilt:

1. n prim $\Rightarrow$ $a^{(n-1)/2} = \pm 1$ für $\forall\, a \in \mathbf{Z}_n \setminus \{0\}$
2. n nicht prim $\Rightarrow$ $a^{(n-1)/2} = \pm 1$ für höchstens die Hälfte der $a \in \mathbf{Z}_n \setminus \{0\}$

(Beweis folgt aus dem Fermatschen Satz, vgl. Übung!)

Primzahlen

- Es existieren **unendlich** viele Primzahlen.
- Es existiert **kein** bekanntes effizientes Verfahren für die Zerlegung großer Zahlen in ihre Primfaktoren.
- Die Zahl 1234567891 ist eine Primzahl.
- Die Zahl 1987654321 ist keine Primzahl, sondern das Produkt von 457 und 4349353.
- $(2^{57885161} - 1)$ ist die größte bekannte Primzahl. Sie hat 17.425.170 Dezimalstellen und wurde 2013 entdeckt (Dr. Cooper, Professor aus Missouri, USA).
- Für jede natürliche Zahl n gibt $\pi(n)$ die Zahl der Primzahlen $\leq n$ an:

$$\pi(n) \approx n / (\ln(n) - 1.08366) \approx n / \ln(n) \quad \text{für großes n.}$$

Primzahlen

Definition: Eine natürliche Zahl n > 1 heißt Primzahl, wenn sie nur durch 1 und sich selbst ohne Rest teilbar ist.

Lösungsalgorithmus:

1. Wir bringen alle Zahlen von 2 bis n in ein Sieb.
2. Wir nehmen die erste (kleinste) Zahl aus dem Sieb heraus und heben sie separat auf. Danach lassen wir alle *nicht-trivialen* Vielfachen dieser Zahl durch das Sieb fallen.
3. Dann nehmen wir die nächst größere der *übriggebliebenen* Zahlen aus dem Sieb heraus und heben auch diese separat auf. Alle Vielfachen dieser Zahl lassen wir wiederum durch das Sieb fallen.
4. Wir wiederholen den 3. Schritt so oft, bis keine Zahlen mehr im Sieb sind.
5. Die separat aufgehobenen Zahlen sind die gesuchten Primzahlen.

Erzeugt Primzahlen `p` unterhalb der Schranke `maxp`.

Pseudocode:

```
for p = 2 bis maxp put Z(p) := TRUE
for p = 2 bis √maxp
   if Z(p) == TRUE then
      {  k = p * p
         do
            Z(k) := FALSE
            k := k + p
         while k <= maxp     }
```

Ausgabe:

∀ Primzahlen `p` = 2 bis `maxp`, für die `Z(p) == TRUE` ist.

Primzahlen zwischen 0 und 300: Rechenbsp.: $n = 300$ $\ln(300) = 5.7$

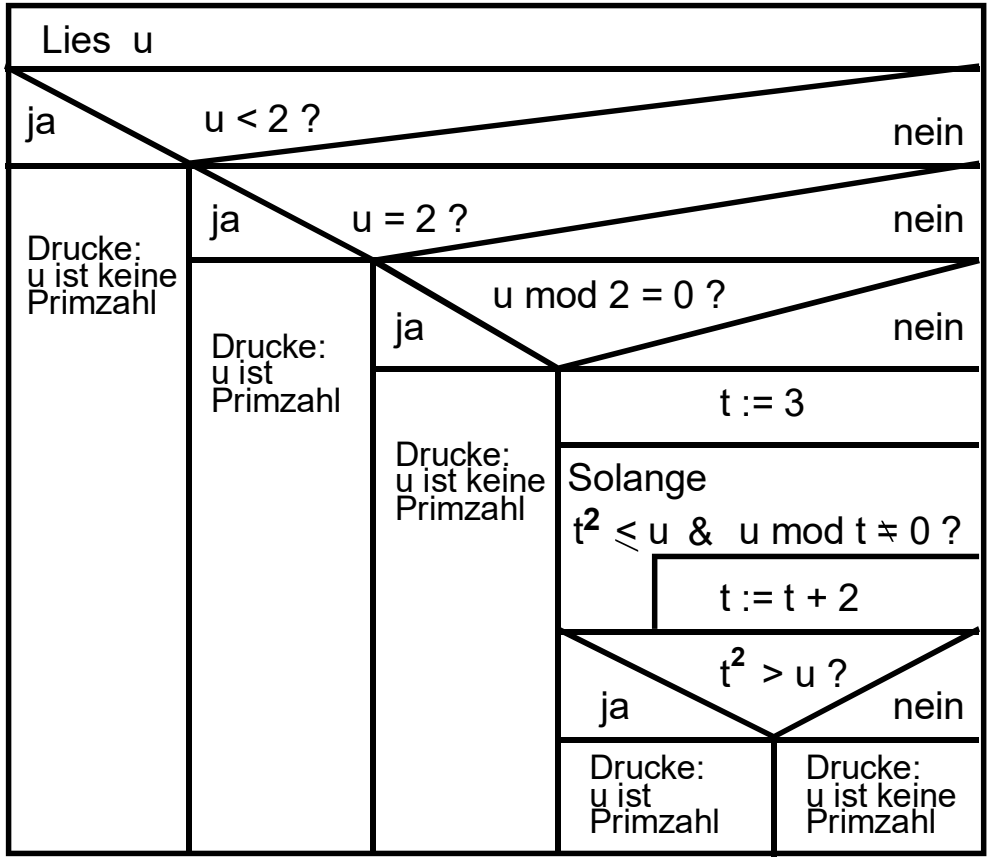
$\Rightarrow$ Es existieren 62 Primzahlen. $\Rightarrow \pi(300) \approx \underline{\underline{64}}$

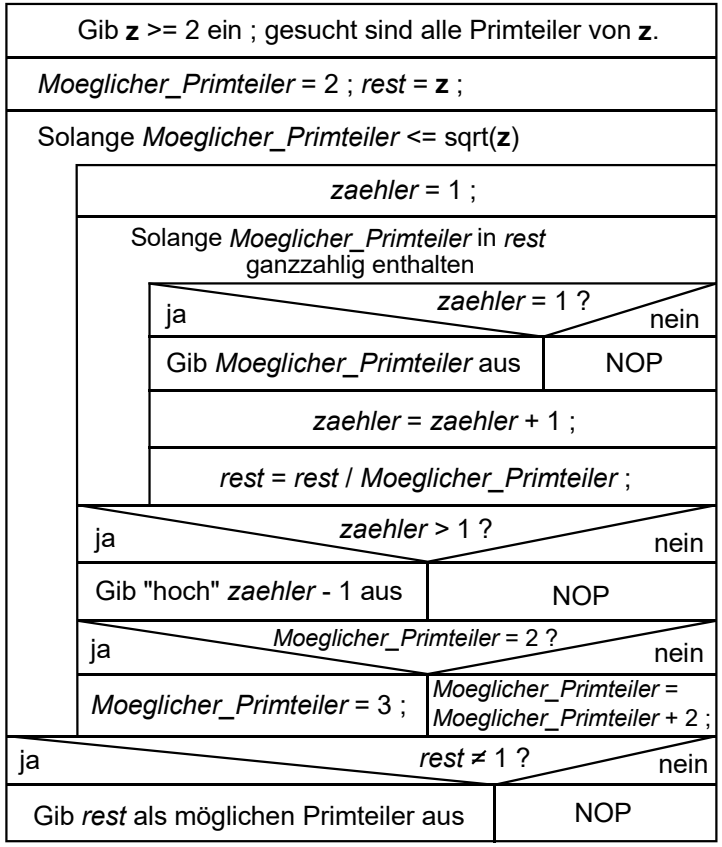
2	3	5	7	11	13	17	19	23	29
31	37	41	43	47	53	59	61	67	71
73	79	83	89	97	101	103	107	109	113
127	131	137	139	149	151	157	163	167	173
179	181	191	193	197	199	211	223	227	229
233	239	241	251	257	263	269	271	277	281
283	293								

Abschätzung:

$$\pi(n) \approx n / (\ln(n) - 1.08366) \approx n / \ln(n) \text{ für großes n.}$$

Primzahlentest und Primteilerzerlegung (deterministisch)





Statistischer Primzahlentest von Miller und Rabin

Sei n eine ungerade Zahl für die auch (n – 1) / 2 ungerade ist (sog. *erlaubte Zahlen*):

1. Wähle Zufallszahlen a_1 , ..., $a_k \in \{1, ..., n-1\}$ aus (auch *Zeugen* genannt).
2. Berechne $a_i^{(n-1)/2}$ in $\mathbf{Z}_n$.
3. Falls alle $a_i^{(n-1)/2} = \pm 1$, entscheide: n ist prim
   sonst entscheide: n ist nicht prim.

Die Fehlerwahrscheinlichkeit des Tests ist $\leq (1/2)^k$, wenn k die Anzahl der gewählten Zeugen ist.

→ Test liefert nicht immer die korrekte Antwort!

Beispiele:

Zahl u	Primzahl ja / nein ?
15.681	nein
194.609	ja
224.711	ja
302.515.449	nein
2.147.483.647	nein

Zahl u	Zerlegung
137.917	$13 \cdot 103^2$
119.394.613	$13^2 \cdot 19^3 \cdot 103$
2 E9	$2^{10} \cdot 5^9$
183.495.637	$13^3 \cdot 17^4$

BBS-Generator

1. Wähle zwei große Primzahlen p und q, die beide bei Division durch 4 den Rest 3 ergeben. D. h.

$$p \equiv q \equiv 3 \bmod 4$$

2. Berechne $n = p \cdot q$ und wähle eine Zufallszahl s, die relativ prim zu n ist. Daraus berechne die Seed-Zahl x_0:

$$x_0 = s^2 \bmod n$$

3. Berechne nun mit i = 1 beginnend wiederholt:

$$x_i = (x_{i-1})^2 \bmod n$$

4. Gib das folgende Bit b_i als i-tes Zufallsbit aus:

$$b_i = x_i \bmod 2 \in \{0, 1\}$$

Zufallszahlengenerierung

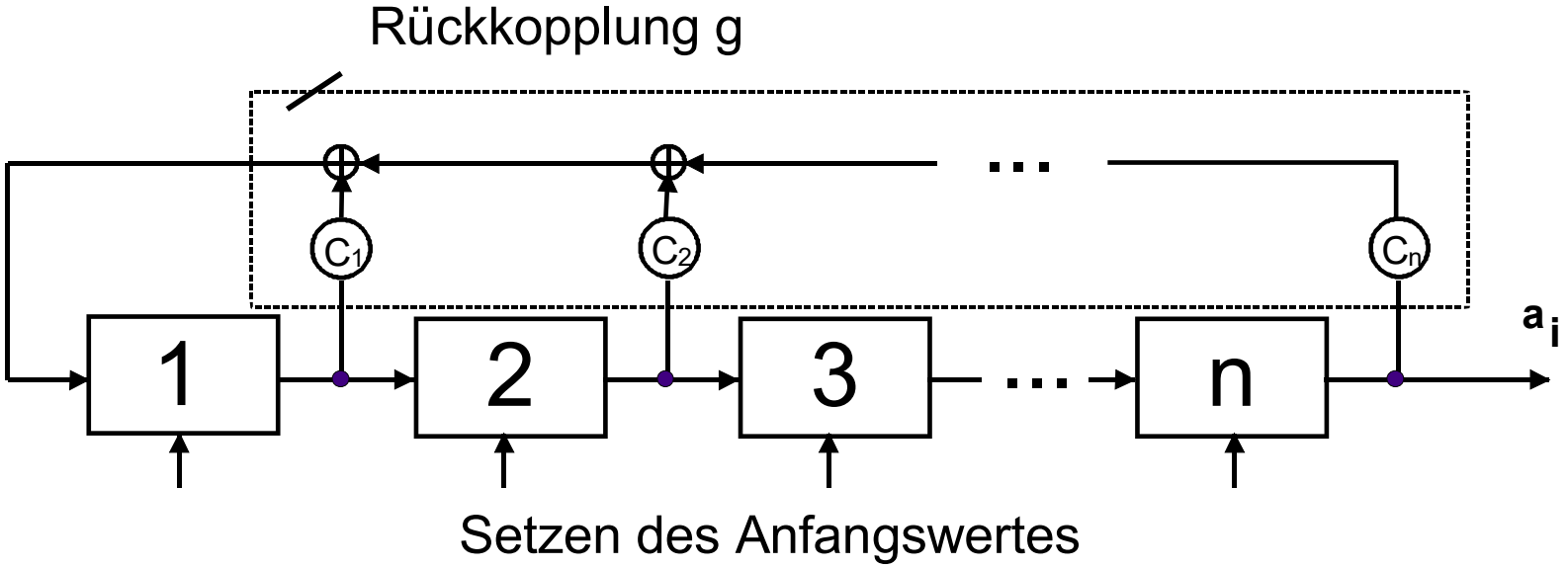
- Zufallszahlen sind ein wesentliches kryptographisches Grundelement.
- Aus ihnen werden Ergebnisse gebildet, die für den bestimmten Anwendungszweck unvorhersehbar sein sollen.
- Sie werden z. B. für die Schlüsselgenerierung, für Initialisierungsvektoren, für Startwerte, etc. benötigt.
- Da die Erzeugung echter Zufallszahlen nicht unproblematisch ist, werden vielerorts Pseudozufallszahlen verwendet.

Zielsetzung:

- Es soll sehr schwierig sein, die Output-Bits eines (Pseudo-) Zufallszahlengenerators zu bestimmen.
- Wenn also n Output-Bits a_1, a_2, ..., a_n vorliegen, soll es rechnerisch unmöglich sein, das (n+1)-te Bit a_{n+1} mit einer Wahrscheinlichkeit, die größer als 0.5 ist, vorherzusagen.

- Je nachdem, ob die Rückkopplungsfunktion g linear oder nichtlinear ist, spricht man von linearen oder nichtlinearen Schieberegistern.
- Die maximale Periode eines n-stufigen Schieberegisters ist 2^n.

- Periodische Zufallsfolgen mit einer möglichst großen Periode bewirken gleichzeitig eine gute statistische Verteilung.
- Die Länge N des kürzesten linear-rückgekoppelten Schieberegisters, durch das die Erzeugung einer vorgegebenen Zufallszahlenfolge a ersetzt werden kann, heißt lineare Komplexität der Folge a.
- Wenn 2 • N aufeinanderfolgende Output-Bits eines N-stufigen linear-rückgekoppelten Schieberegisters bekannt sind, können alle folgenden Output-Bits vorausgesagt werden.



g = Rückkopplungsfunktion
n = Anzahl der Stufen
$C_i \in 0 \vee 1$; je nachdem, ob entspr. Rückführung vorhanden
$\oplus$ = XOR-Verknüpfung bzw. modulo-2-Addition

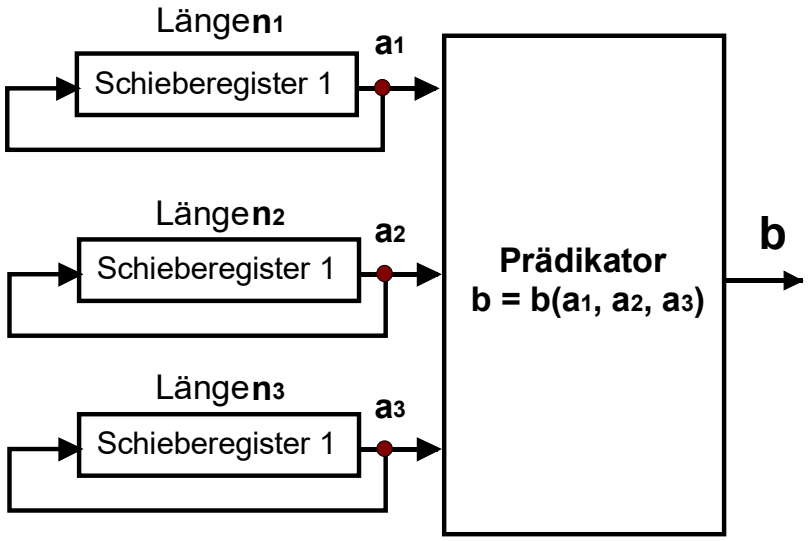
Der Geffe-Generator besteht aus drei zueinander primitiven, nicht-linear-rückgekoppelten Schieberegistern der Länge n_1, n_2 und n_3, d. h. ihre Produktdarstellungen enthalten keine gemeinsamen Faktoren.

Periode P:

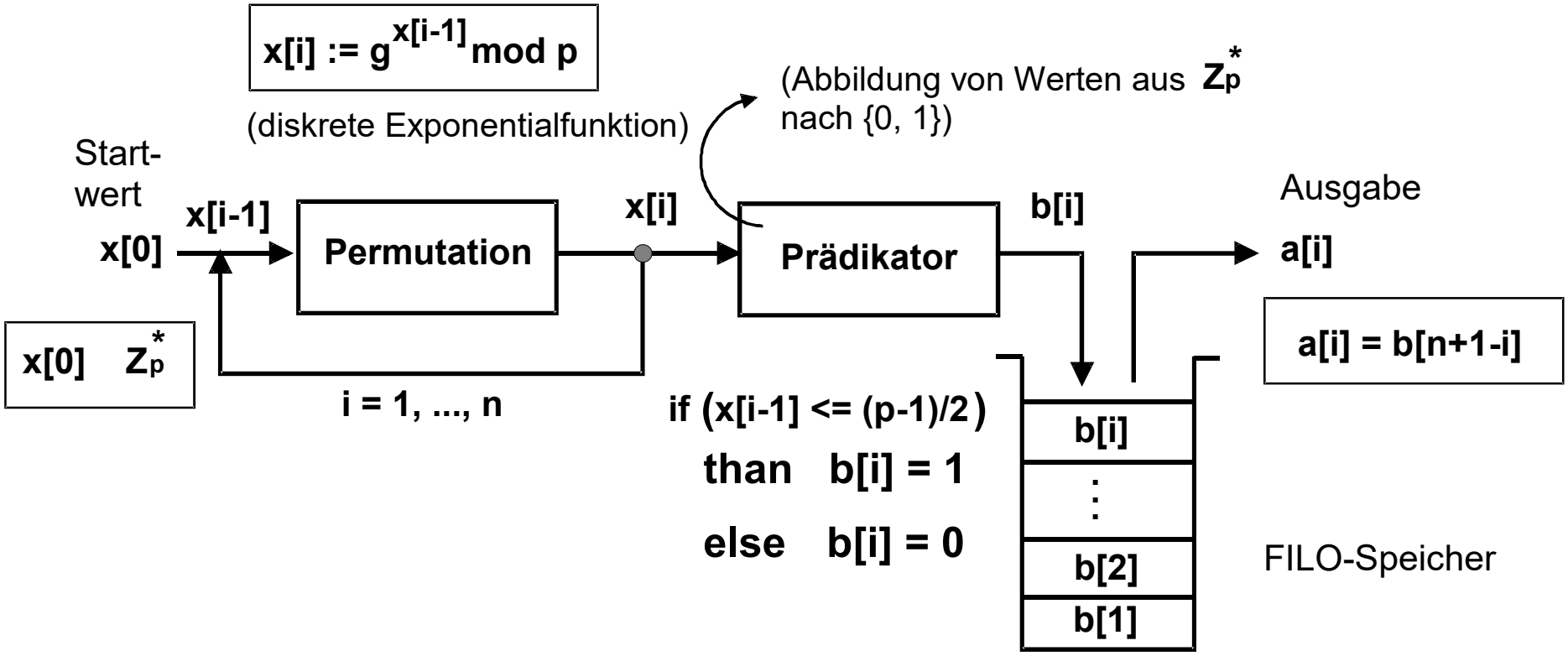
$$P = kgV(2^{n_1} - 1, 2^{n_2}-1, 2^{n_3}-1)$$

Lineare Komplexität N:

$$N = n_3 * n_1 + (n_1 + 1) * n_2$$



Prädikator: $b = a_3 \oplus \overline{a_1} * (a_2 \oplus a_3)$


x[i] := g^x[i-1] mod p
(diskrete Exponentialfunktion)
(Abbildung von Werten aus Z*p nach {0, 1})
Start-wert
x[0]
x[i-1]
Permutation
x[i]
Prädikator
b[i]
Ausgabe
a[i]
a[i] = b[n+1-i]
x[0] Z*p
i = 1, ..., n
if (x[i-1] <= (p-1)/2)
than b[i] = 1
else b[i] = 0
b[i]
b[2]
b[1]
FILO-Speicher

Notation:

- p (Modulus) sei eine Primzahl; $p \in \mathbf{P}$
- n ist die Ausgabelänge; $n \in \mathbf{N}$
- $\mathbf{Z_n^*}$ ist eine **multiplikative Gruppe** (des Restklassenrings $\mathbf{Z_n}$) bestehend aus den Elementen von $\mathbf{Z_n}$, die zu n teilerfremd sind.

$$\mathbf{Z_n^*} := \{a \in \mathbf{Z_n} \setminus \{0\} \mid ggT(a, n) = 1\}$$

- g sei **Primitivwurzel** aus $\mathbf{Z_p^*}$; Zahl g sollte so gewählt werden, dass sie eine möglchst große Untergruppe von $\mathbf{Z_p}$ erzeugt (vgl. DL-Problem).
- X[0] ist Saat aus $\mathbf{Z_p^*}$

Ablauf:

```
var     x[0:n]    : array of integer ;
        b[1:n]    : array of {0, 1} ;

for i = 1 to n do
    begin                                          /* Berechne nächsten Zwischen- */
        x[i] = g^x[i–1] mod p ;                    /* wert durch Permutation von x[i-1] */
        if (x[i–1] ≤ (p – 1)/2) then  b[i] = 1     /* Berechne das Prädikat mit */
                                else  b[i] = 0     /* Hilfe von x[i - 1] */
    end ;
```

Output:

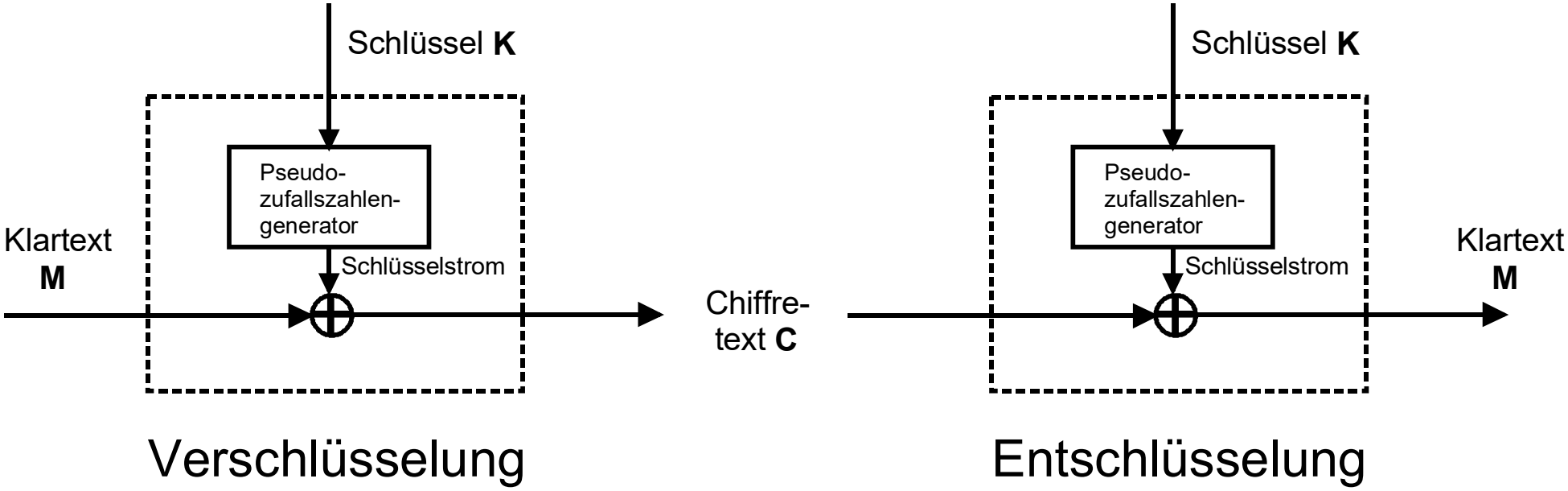
```
for i = 1 to n do
    begin
        a[i] = b[n + 1 – i] ;                      /* Ausgabe in umgekehrter Reihenfolge */
        output a[i] ;
    end ;
```

Typische **Beurteilungskriterien** sind:

- Periode P
- Lineare Komplexität N
- Statistische Eigenschaften wie
  - Häufigkeitsverteilung von Bits und Bitgruppen
  - Korrelation zwischen Bitfolgen (z. B. zwischen Klar- und Schlüsseltexten)
  - Mittelwerttests
  - Abstände zwischen dem zweimaligen Auftreten eines Bitmusters
  - uvm.

Bitstromverschlüsselung

- Symmetrische Verschlüsselungsverfahren können in Blockalgorithmen und Bitstrom-verschlüsselung unterschieden werden.
- Bei der Bitstromverschlüsselung wird der Klartext Bit-für-Bit verschlüsselt.
- Der angewandte geheime Schlüssel(strom) wird in einem Pseudo-Zufallszahlen-generator eingegeben oder der Anfangszustand des Pseudo-Zufallszahlengenerators wird mit dem geheimen Schlüssel vorbelegt (Initialisierung).
- Die Periode der vom Pseudo-Zufallszahlengenerator erzeugten Folge muß größer als die zu verschlüsselnde Nachricht sein (sog. one time pat).
- Die Output-Bits werden auf der Verschlüsselungsseite mit dem Klartext durch XOR verknüpft.
- Auf der Entschlüsselungsseite wird der Pseudo-Zufallszahlengenerator mit demselben geheimen Schlüssel vorbelegt, und die Output-Folge mit dem Schlüsseltext wiederum durch XOR verknüpft, so daß man wieder den ursprünglichen Klartext erhält.
- Ein besonderes Problem stellt der Austausch des geheimen Schlüssels dar.


Schlüssel K
Pseudo-
zufallszahlen-
generator
Schlüsselstrom
Klartext
M
Chiffre-
text C
Schlüssel K
Pseudo-
zufallszahlen-
generator
Schlüsselstrom
Klartext
M
Verschlüsselung
Entschlüsselung

```
/* Datum: 19.07.2002                                              */
/* Autor: Bernhard Geib                                           */
/* Funktion: Verschluesselung mit Coset-Muster 01010101           */

#include <stdio.h>

int main (void)
{  int c;

   c = getchar();
   while (c != EOF)                {
      if (c != '\n')
      {
         c = c ^ 85;    /* hier eine Zufallszahl verwenden */
      }
      printf ("%c", c);
      c = getchar();               }

   return 0;                       }
```

```
/* Datum: 17.07.2002                                  */
/* Autor: Bernhard Geib                               */
/* Funktion: Bijektive, selbstinverse Chiffre         */

#include <stdio.h>
#include <string.h>


void verschluessler(char *text, const char *geheimnis)
{
   /* Annahme: text und geheimnis zeigen jeweils auf   */
   /* string mit Zeichen aus dem Bereich 32 .. 95      */

      int i ;
      int lt = strlen(text) ;
      int lg = strlen(geheimnis) ;
```

```
// Fortsetzung Verschluessler

     for (i=0; i < lt; ++i)
     {
       char c = text[i] - ' ' ;
       char key = geheimnis[i % lg] - ' ' ;

       c = c ^ key ;
       text[i] = c + ' ' ;
     }

} // Ende Verschluessler
```

```
// Zugehoeriges Hauptprogramm

int main( void )
{
        char text[] = "Dies ist der gegebene Klartext" ;
        char geheimnis[] = "GEHEIM" ;

        printf("Vor  der Verschlüsselung: %s\n", text) ;
        verschluessler(text, geheimnis) ;
        printf("Nach der Verschlüsselung: %s\n", text) ;
        verschluessler(text, geheimnis) ;
        printf("Nach der Entschlüsselung: %s\n", text) ;

        return 0 ;
}
```

Zusammenfassung

- Gegenstand der elementaren Zahlentheorie sind die natürlichen Zahlen **N** und deren Erweiterungen **Z** (ganze Zahlen), sowie der Körper **Q** der rationalen Zahlen.
- Der Gebrauch der Zahlentheorie setzt an manchen Stellen auch grundlegende Kenntnisse der Algebra (Gruppen- und Ringtheorie) sowie der linearen Algebra voraus.
- Das Material zur Lehrveranstaltung ist so ausführlich wie nötig und so knapp wie möglich zusammengestellt, um hier sämtliche Grundlagen der folgenden Praktikumsaufgaben zu legen und doch das erste Verständnis nicht zu erschweren.
- In einer Krypto-Library wurden die für unsere Zwecke effizientesten Algorithmen (erweiterter Euklidischer Algorithmus, Square & Multiply Algorithmus, Primalitäts- und Faktorisierungsalgorithmus, Chinesischer Restalgorithmus etc.) implementiert.
- Damit werden die Teilnehmer in die Lage versetzt, praktische Kryptoverfahren nachprogrammieren und den Aufwand von insbesondere asymmetrischen Verschlüsselungs-, Signatur- und Authentifikationssystemen abschätzen zu können.