Einführung in die Zahlentheorie

Dozent: Prof. Dr. Michael Eichberg

Kontakt: michael.eichberg@dhbw.de

Version: 1.1.5

Quelle: Im Wesentlichen *Cryptography and Network Security - Principles and Practice, 8th Edition, William Stallings*

Folien: https://delors.github.io/sec-einfuehrung-in-die-zahlentheorie/folien.de.rst.html https://delors.github.io/sec-einfuehrung-in-die-zahlentheorie/folien.de.rst.html.pdf

Fehler melden: https://github.com/Delors/delors.github.io/issues

1. Teilbarkeit

Teilbarkeit

- Ein b ungleich Null teilt a wenn $a = mb$ für ein beliebiges m und a, b und m ganze Zahlen sind.
- b teilt a wenn es keinen Rest bei der Division gibt.
- Die Notation $b|a$ bedeutet, dass b a teilt.
- Wenn $b|a$ gilt, dann sagen wir auch, dass b ein Teiler von a ist.

Beispiel

Die positiven Teiler von 24 sind: 1, 2, 3, 4, 6, 8, 12 und 24.

Weitere Beispiele: $13|182$; $-5|30$; $17|289$; $-3|33$; $17|0$.

Eigenschaften der Teilbarkeit

- Wenn $a|1$, dann gilt $a = \pm 1$.
- Wenn $a|b$ und $b|a$, dann gilt $a = \pm b$.
- Jedes $b \neq 0$ teilt 0.
- Wenn $a|b$ und $b|c$, dann $a|c$.

Beispiel

Sei $a = 11$, $b = 66$ und $c = 198$, dann gilt:

$11|66$ und $66|198 \Rightarrow 11|198$

Wenn $b|g$ und $b|h$, dann $b|(mg + nh)$ für beliebige ganze Zahlen m und n.

Beispiel

$3|27$ und $3|33 \Rightarrow 3|(m \times 27 + n \times 33)$

Beweis

Wenn $b|g$, dann gilt für g, dass $g = b \times g_1$ ist für eine beliebige ganze Zahl g_1.

Wenn $b|h$, dann gilt für h, dass $h = b \times h_1$ ist für eine beliebige ganze Zahl h_1.

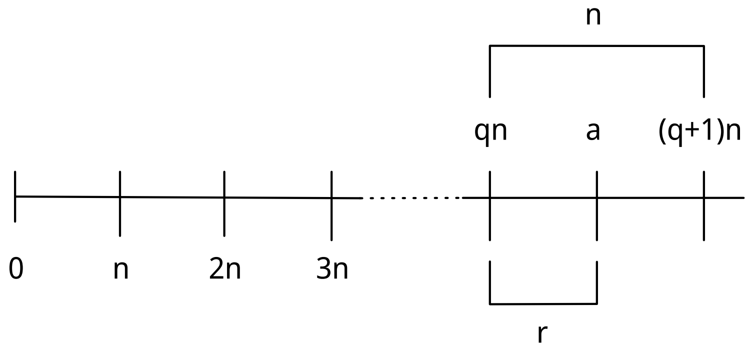
Somit gilt: $mg + nh = mbg_1 + nbh_1 = b \times (mg_1 + nh_1)$ und deshalb wird $mg+nh$ von b geteilt. ∎

Teilungsalgorithmus

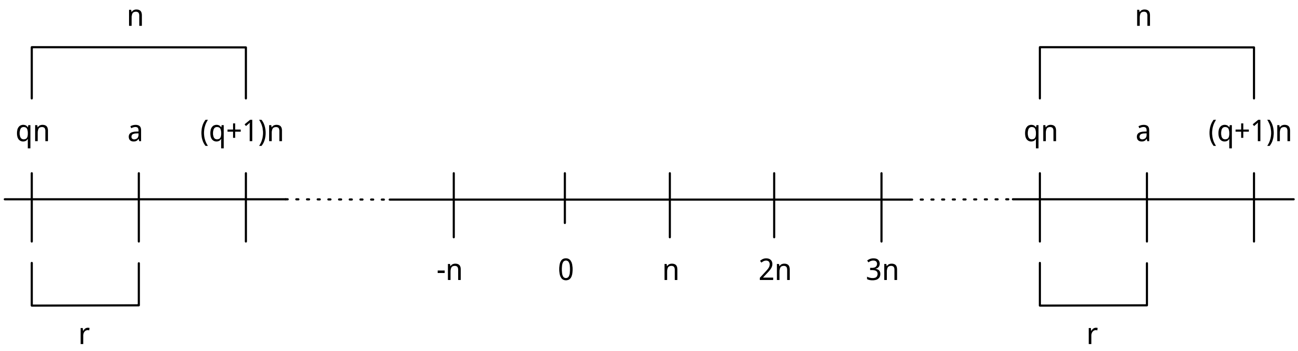
Sei eine beliebige positive ganze Zahl n gegeben und eine beliebige ganze Zahl $a \geq 0$, so erhält man bei der Division von a durch n:

- einen ganzzahligen Quotienten $q \in \mathbb{Z}$ und
- einen nicht negativen, ganzzahligen Rest $r \in \mathbb{N}_0$,

die der folgenden Beziehung gehorchen: $a = qn + r; \quad 0 \leq r < n, \quad q = \lfloor a/n \rfloor$



Teilungsalgorithmus für negative a



Beispiel

$$a = -11; n = 7; -11 = (-2) \times 7 + 3; \quad r = 3 \quad q = -2$$

Euklidischer Algorithmus

Eine der grundlegenden Techniken der Zahlentheorie.

Verfahren zur Bestimmung des größten gemeinsamen Teilers (GGT) von zwei positiven ganzen Zahlen.

Definition

Zwei ganze Zahlen sind **relativ prim** (*relatively prime*), wenn ihr einziger gemeinsamer positiver ganzzahliger Faktor 1 ist (z. B. 7 und 9, aber auch 3 und 8).

Größter Gemeinsamer Teiler (GGT)

(🇺🇸 *Greatest Common Divisor (GCD)*)

- Der größte gemeinsame Teiler von zwei ganzen Zahlen a und b ist die größte ganze Zahl, die sowohl a als auch b teilt.
- Wir verwenden die Schreibweise $ggt(a, b)$ für den GGT von a und b.
- Wir definieren $ggt(0, 0) = 0$.
- Die **positive** ganze Zahl c wird als GGT von a und b bezeichnet, wenn:
  - c ein Teiler von a und b ist
  - jeder Teiler von a und b ein Teiler von c ist

Alternative Definition des GGT

$$ggt(a,b) = max[k, \text{so dass } k|a \text{ und } k|b]$$

Beispiel

$ggt(60, 24) =$

$ggt(60, -24) =$

12

GGT und „relativ prim“

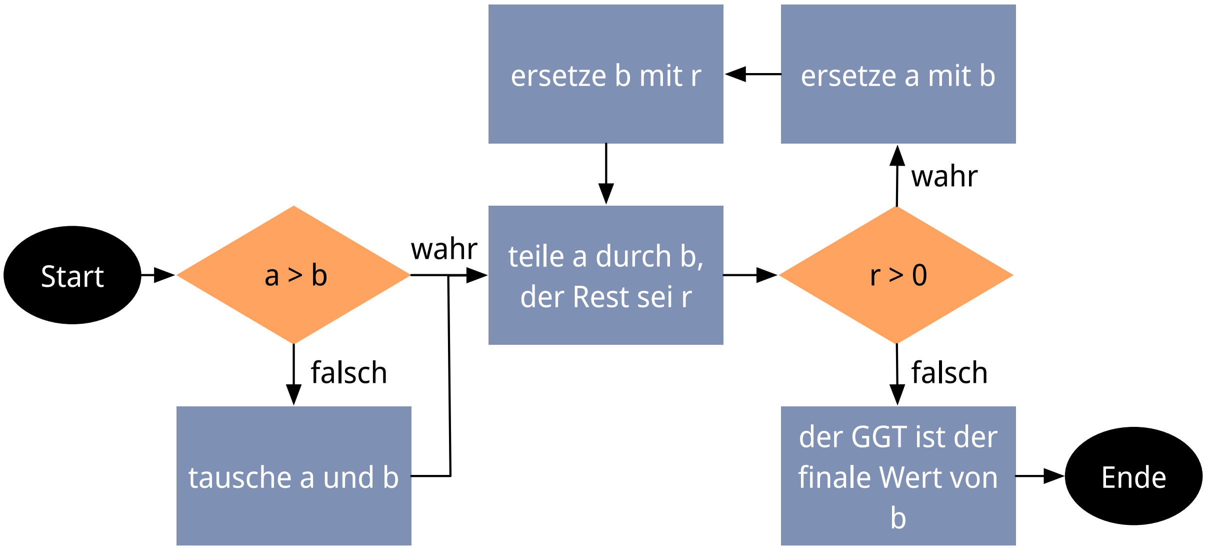
Beobachtung

Zwei ganze Zahlen a und b sind relativ prim, wenn ihr einziger gemeinsamer positiver ganzzahliger Faktor 1 ist.

$$\Leftrightarrow$$

a und b sind relativ prim wenn $ggt(a, b) = 1$

Berechnung des GGT ($ggt(a, b)$) mit Hilfe des euklidischen Algorithmus



Beispiel für die Berechnung des GGT ($ggt(710, 310)$) mit Hilfe des euklidischen Algorithmus

$$
\begin{aligned}
710 &= 2 \cdot 310 + 90 \\
310 &= 3 \cdot 90 + 40 \\
90 &= 2 \cdot 40 + 10 \\
40 &= 4 \cdot 10 + 0
\end{aligned}
$$

Euklidischer Algorithmus

$ggt(1.160.718.174, 316.258.250)$

Schritt	Dividend	Divisor	Quotient	Rest
1	1.160.718.174	316.258.250	3	211.943.424
2	316.258.250	211.943.424	1	104.314.826
3	211.943.424	104.314.826	2	3.313.772
4	104.314.826	3.313.772	31	1.587.894
5	3.313.772	1.587.894	2	137.984
6	1.587.894	137.984	11	70.070
7	137.984	70.070	1	67.914
8	70.070	67.914	1	2.156
9	67.914	2.156	31	1.078
10	2.156	**1.078**	2	0

2. Modulare Arithmetik

Der Modulus

Wenn a eine ganze Zahl und n eine positive ganze Zahl ist, dann definieren wir $a \bmod n$ als Rest der Division von a durch n. Die ganze Zahl n wird als Modulus bezeichnet.

Somit gilt für jede ganze Zahl a:

$$a = qn + r \quad 0 \leq r < n; \qquad q = \lfloor a/n \rfloor$$
$$a = \lfloor a/n \rfloor \times n + (a \bmod n)$$

Beispiel

$$11 \bmod 7 = 4; \qquad -11 \bmod 7 = 3$$

Modulare Arithmetik (*kongruent modulo* n)

- Zwei ganze Zahlen a und b werden als *kongruent modulo* n bezeichnet, wenn $(a \bmod n) = (b \bmod n)$
- Wir verwenden die Schreibweise $a \equiv b \pmod{n}$.
- Beachten Sie, dass, wenn $a \equiv 0 \pmod{n}$ ist, dann gilt $n|a$.

Beispiel

$73 \equiv 4 \pmod{23}; \qquad 21 \equiv -9 \pmod{10}; \qquad 81 \equiv 0 \pmod{27}$

Hinweis

Der Operator mod wird ...

a. als binärer Operator verwendet, der einen Rest erzeugt, und
b. als Kongruenzrelation, die die Gleichwertigkeit zweier ganzer Zahlen anzeigt.

Hinweis:

$21 \equiv -9 \pmod{10} \Leftrightarrow 21 \bmod 10 = -9 \bmod 10 = 1$

$-9 \mod 10 \Leftrightarrow -9 = n * 10 + 1$

Eigenschaften der Kongruenz

1. $a \equiv b \pmod{n}$ wenn $n|(a-b)$ (Siehe nächste Folie.)
2. $a \equiv b \pmod{n} \Rightarrow b \equiv a \pmod{n}$
3. $a \equiv b \pmod{n}$ und $b \equiv c \pmod{n} \Rightarrow a \equiv c \pmod{n}$

$a \equiv b \pmod{n}$ wenn $n|(a-b)$ — Erklärt

Wenn $n|(a-b)$, dann gilt $(a-b) = kn$ für ein k

- Wir können also schreiben $a = b + kn$.
- Deshalb gilt $(a \bmod n) = ((b+kn) \bmod n) =$

  Rest wenn $b + kn$ geteilt wird durch $n =$

  Rest wenn b geteilt wird durch $n =$

  $(b \bmod n)$

Beispiel

$23 \equiv 8 \pmod{5}$, da $23 - 8 = 15 = 5 \times 3$

$-11 \equiv 5 \pmod{8}$, da $-11 - 5 = -16 = 8 \times -2$

$81 \equiv 0 \pmod{27}$, da $81 - 0 = 81 = 27 \times 3$

Im zweiten Schritt haben wir $\bmod n$ angewendet.

Eigenschaften der modularen Arithmetik

1. $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$
2. $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$
3. $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

$[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$ — Erklärt

Definiere $(a \bmod n) = r_a$ und $(b \bmod n) = r_b$.

Dann können wir:

- $a = r_a + jn$ für eine ganze Zahl j und
- $b = r_b + kn$ für eine ganze Zahl k schreiben.

Dann gilt:

$$\begin{aligned}(a + b) \bmod n &= (r_a + jn + r_b + kn) \bmod n \\ &= (r_a + r_b + (k + j)n) \bmod n \\ &= (r_a + r_b) \bmod n \\ &= [(a \bmod n) + (b \bmod n)] \bmod n\end{aligned}$$

Im vorletzten Schritt setzen wir die Definition vom Anfang ein und erhalten das Ergebnis.

Modulare Arithmetik (Beispiele für Eigenschaften)

Beispiel

$$11 \bmod 8 = 3; \qquad 15 \bmod 8 = 7$$

- $$[(11 \bmod 8) + (15 \bmod 8)] \bmod 8 = [3 + 7] \bmod 8 = 10 \bmod 8 = 2$$
  $$(11 + 15) \bmod 8 = 26 \bmod 8 = 2$$
- $$[(11 \bmod 8) - (15 \bmod 8)] \bmod 8 = [3 - 7] \bmod 8 = -4 \bmod 8 = 4$$
  $$(11 - 15) \bmod 8 = -4 \bmod 8 = 4$$
- $$[(11 \bmod 8) \times (15 \bmod 8)] \bmod 8 = [3 \times 7] \bmod 8 = 21 \bmod 8 = 5$$
  $$(11 \times 15) \bmod 8 = 165 \bmod 8 = 5$$

Rechnung Modulo 8

Definition

$$Z_n = \{0, 1, \ldots, (n-1)\}$$

Beispiel

$$Z_8 = \{0, 1, 2, 3, 4, 5, 6, 7\}$$

Addition

$+$	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	*0*	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	*0*
2	2	3	4	5	6	7	*0*	1
3	3	4	5	6	7	*0*	1	2
4	4	5	6	7	*0*	1	2	3
5	5	6	7	*0*	1	2	3	4
6	6	7	*0*	1	2	3	4	5
7	7	*0*	1	2	3	4	5	6

Multiplikation

$\times$	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0	0	0	0	0	0	0	0
1	0	*1*	2	3	4	5	6	7
2	0	2	4	6	0	2	4	6
3	0	3	6	*1*	4	7	2	5
4	0	4	0	4	0	4	0	4
5	0	5	2	7	4	*1*	6	3
6	0	6	4	2	0	6	4	2
7	0	7	6	5	4	3	2	*1*

Generator in Python:

```python
for i in range(0,8):
    print(str(i)+", ",end="")
    for j in range(0,8):
        v = (i*j) % 8
        if v == 1:
            v = "*"+str(v)+"*"
        else:
            v = str(v)
        print(v+",",end="")
    print()
```

Additive und Multiplikative Inverse Modulo 8

Definition

Die **negative/additive Inverse** einer ganzen Zahl x ist die ganze Zahl y, für die gilt: $(x + y) \bmod 8 = 0$.

Die **muliplikative Inverse** einer ganzen Zahl x ist die ganze Zahl y, für die gilt: $(x \times y) \bmod 8 = 1$.

w	$-w$	w^{-1}
0	0	—
1	7	1
2	6	—
3	5	3
4	4	—
5	3	5
6	2	—
7	1	7

Eigenschaften der modularen Arithmetik für ganze Zahlen in Z_n

Kommutativgesetz:	$(w + x) \bmod n = (x + w) \bmod n$ $(w \times x) \bmod n = (x \times w) \bmod n$
Assoziativgesetz:	$[(w + x) + y] \bmod n = [w + (x + y)] \bmod n$ $[(w \times x) \times y] \bmod n = [w \times (x \times y)] \bmod n$
Distributivgesetz:	$[w \times (x + y)] \bmod n = [(w \times x) + (w \times y)] \bmod n$
Identitäten:	$(0 + w) \bmod n = w \bmod n$ $(1 \times w) \bmod n = w \bmod n$
Additive Inverse (-w):	Für jedes $w \in Z_n$ gibt es ein z, so dass $w + z \equiv 0 \pmod{n}$

Euklidischer Algorithmus - neu betrachtet

Satz

Für beliebige ganze Zahlen a und b mit $a \geq b \geq 0$,

$$ggt(a, b) = ggt(b, a \bmod b)$$

Algorithmus

```
def Euclid(a,b):
    if (b = 0) then
        return a;
    else
        return Euclid(b, a mod b);
```

Beispiel

```
ggt(10,6)
   ↳ ggt(6,4)
      ↳ ggt(4,2)
         ↳ ggt(2,0)
2            ↩
```

? Frage

Um welche Art von rekursivem Algorithmus handelt es sich hierbei?

In der gegebenen Formulierung ist der Algorithmus endrekursiv (*tail recursive*).

Erweiterter Euklidischer Algorithmus

- Erforderlich für Berechnungen im Bereich der endlichen Körper und Verschlüsselungsalgorithmen wie RSA.
- Für zwei ganze Zahlen a und b berechnet der erweiterte euklidische Algorithmus den GGT d, aber auch zwei zusätzliche ganze Zahlen x und y, die die folgende Gleichung erfüllen:

$$x \times a + y \times b = d = ggt(a, b)$$

Notwendigerweise haben x und y gegensätzliche Vorzeichen, da sonst $(x \times a + y \times b) > a \ (> b)$ gelten würde und somit nicht den GGT darstellen könnte.

Der erweiterte euklidische Algorithmus kann auf jeden Ring angewandt werden, in welchem eine Division mit kleinstem Rest durchgeführt werden kann. Ein Beispiel ist der Polynomring in einer Variablen mit rationalen oder reellen Koeffizienten wie sie bei der Verschlüsselung angewandt werden. Wir werden dies später wieder aufgreifen.

Der erweiterte Algo. dient insbesondere der Berechnung der inversen Elemente in ganzzahligen Restklassenringen. (Beides werden wir später in der Vorlesung betrachten).

$ggt(a = 42, b = 30)$ mit erweitertem Euklidischen Algorithmus

Werfen wir zuerst einen Blick auf $x \times a + y \times b$ für einige x und y:

y \ x	-3	-2	-1	0	1	2	3
-3	-216	-174	-132	-90	-48	-6	36
-2	-186	-144	-102	-60	-18	24	66
-1	-156	-114	-72	-30	12	54	96
0	-126	-84	-42	0	42	84	126
1	-96	-54	-12	30	72	114	156
2	-66	-24	18	60	102	144	186
3	-36	6	48	90	132	174	216

Hinweis

Der GGT von 42 und 30 ist 6 und erscheint in der Tabelle ($x = -2$ und $y = 3$).

Erweiterter Euklidischer Algorithmus - Systematische Berechnung für $ggt(710, 310)$

Klassischer Algorithmus:

```
710 = 2 · 310  + 90
310 = 3 ·  90  + 40
 90 = 2 ·  40  + 10
 40 = 4 ·  10  + 0
```

Umgestellt:

```
710 - 2 · 310 = 90
310 - 3 · 90  = 40
 90 - 2 · 40  = 10
```

Aufgrund der Umstellung z. B. von $710 = 2 \times 310 + 90$ nach $90 = 710 - 2 \times 310$ können wir dann im nächsten Schritt/der nächsten Formel die 90 durch $710 - 2 \times 310$ ersetzen und werden dann $310 - 3 \times (710 - 2 \times 310) = 40$ erhalten.

D. h. betrachten wir 710 als a und 310 als b, erhalten wir durch die Umstellung eine Formel, die nur aus as und bs zuzüglich zweiter Koeffizienten und dem Rest r besteht.

```
                        710 - 2 ·310 = 90
                                       =
               310 - 3 · (710 - 2 ·310) = 40
                                          =
710 - 2 ·310 - 2 · (310 - 3 · (710 - 2 ·310)) = 10
     710 - 2 ·310 - 2 ·310 + 6 · (710 - 2 ·310) = 10
          710 - 2 ·310 - 2 ·310 + 6 ·710 - 12 ·310 = 10
                                  7 ·710 - 16 ·310 = 10
```

$\Rightarrow x = 7$ und $y = -16$

Erweiterter Euklidischer Algorithmus - Formeln

Wir nehmen an, dass wir bei jedem Schritt i die ganzen Zahlen x_i und y_i finden können, die folgende Bedingung erfüllen: $r_i = ax_i + by_i$.

$$\begin{array}{cc}
Original & Erweiterung \\
a = q_1 b + r_1 & r_1 = ax_1 + by_1 \\
b = q_2 r_1 + r_2 & r_2 = ax_2 + by_2 \\
r_1 = q_3 r_2 + r_3 & r_3 = ax_3 + by_3 \\
\vdots & \vdots \\
r_{n-2} = q_n r_{n-1} + r_n & r_n = ax_n + by_n \\
r_{n-1} = q_{n+1} r_n + 0 & \\
d = ggt(a, b) = r_n &
\end{array}$$

Erweiterter Euklidischer Algorithmus

Berechne	Was erfüllt	Berechne	Was erfüllt
$r_{-1} = a$		$x_{-1} = 1;\ y_{-1} = 0$	$a = ax_{-1} + by_{-1}$
$r_0 = b$		$x_0 = 0;\ y_0 = 1$	$b = ax_0 + by_0$
$r_1 = a \bmod b;\ q_1 = \lfloor a/b \rfloor$	$a = q_1 b + r_1$	$x_1 = x_{-1} - q_1 x_0 = 1;$ $y_1 = y_{-1} - q_1 y_0 = -q_1$	$r_1 = ax_1 + by_1$
$r_2 = b \bmod r_1;\ q_2 = \lfloor b/r_1 \rfloor$	$b = q_2 r_1 + r_2$	$x_2 = x_0 - q_2 x_1;\ y_2 = y_0 - q_2 y_1$	$r_2 = ax_2 + by_2$
$r_3 = r_1 \bmod r_2;\ q_3 = \lfloor r_1/r_2 \rfloor$	$r_1 = q_3 r_2 + r_3$	$x_3 = x_1 - q_3 x_2;\ y_3 = y_1 - q_3 y_2$	$r_3 = ax_3 + by_3$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$r_n = r_{n-2} \bmod r_{n-1}$ $; q_n = \lfloor r_{n-2}/r_{n-1} \rfloor$	$r_{n-2} = q_n r_{n-1} + r_n$	$x_n = x_{n-2} - q_n x_{n-1}$ $; y_n = y_{n-2} - q_n y_{n-1}$	$r_n = ax_n + by_n$
$r_{n+1} = r_{n-1} \bmod r_n = 0$ $; q_{n+1} = \lfloor r_{n-1}/r_n \rfloor$	$r_{n-1} = q_{n+1} r_n + 0$		

Lösung

$$d = ggt(a,b) = r_n; x = x_n; y = y_n$$

Erweiterter Euklidischer Algorithmus - Beispiel $ggt(1759, 550)$

i	r_i	q_i	x_i	y_i
-1	1759		1	0
0	550		0	1
1	109	3	1	-3
2	5	5	-5	16
3	4	21	106	-339
4	1	1	-111	355
5	0	4		

Resultat: $d = 1; x = -111; y = 355$

3. Primzahlen und Primzahlenbestimmung

Primzahlen

- Primzahlen haben als Teiler nur 1 und sich selbst.
- Sie können nicht als Produkt von anderen Zahlen geschrieben werden.
- Jede ganze Zahl $a > 1$ kann auf eindeutige Weise faktorisiert werden als: $a = p_1^{a_1} \times p_2^{a_2} \times \ldots \times p_t^{a_t}$ wobei $p_1 < p_2 < \ldots < p_t$ Primzahlen sind und wobei jedes a_i eine positive ganze Zahl ist.

  $$a = \prod_{p \in P} p^{a_p} \quad \text{wenn } a_p \geq 0$$
- Dies ist als Fundamentalsatz der Arithmetik bekannt.

Beispiel

$15 = 2^0 \times 3^1 \times 5^1$

$50 = 2^1 \times 3^0 \times 5^2$

$60 = 2^2 \times 3^1 \times 5^1$

Primzahlen spielen in der Zahlentheorie eine zentrale Rolle. Wir betrachten sie hier aber nur insoweit es für das Verständnis der Kryptographie notwendig ist.

Die Zerlegung zu bestimmen geschieht dadurch, dass man die Zahl durch die kleinste Primzahl (2) solange teilt, bis dies nicht mehr ohne Rest möglich ist, die Potenz ergibt sich dann aus der Anzahl der erfolgreichen Teilungen. Danach fährt man mit der nächsten Primzahl fort (3, 5, 7, ...) bis die Zahl zerlegt wurde.

Fermats (kleines) Theorem

Besagt folgendes:

- Wenn p eine Primzahl und a eine positive ganze Zahl ist, die nicht durch p teilbar ist (d.h. $p \nmid a$), dann gilt
$a^{p-1} \equiv 1(mod\ p)$

Bemerkung

Wichtig in der Public-Key-Kryptographie.

Alternative form:

- Wenn p eine Primzahl und a eine positive ganze Zahl ist, dann ist $a^p \equiv a \pmod p$

Beispiel

$$\text{Sei } p = 7 \text{ und } a = 2: \ (2^6 = 64) \equiv 1 \pmod 7, \quad da\ 64/7 = 9\ Rest\ 1$$
$$\text{Sei } p = 7 \text{ und } a = 8: \ (8^6 = 262.144) \equiv 1 \pmod 7$$

Mit anderen Worten: a ist kein vielfaches von p.

Herleitung der alternativen Form:

$$\begin{aligned}
a^{p-1} \bmod p &= 1 \bmod p && \mid \times a \bmod p \\
a^{p-1} \bmod p \times a \bmod p &= a \bmod p && \mid \bmod p \\
(a^{p-1} \bmod p \times a \bmod p) \bmod p &= (a \bmod p) \bmod p \\
(a^{p-1} \times a) \bmod p &= a \bmod p \\
a^p \bmod p &= a \bmod p
\end{aligned}$$

Die Eulersche Totientenfunktion $\phi(n)$

Definition

$\phi(n)$ gibt die Anzahl der positiven ganzen Zahlen, die kleiner als n und relativ prim zu n sind an.

Per Konvention ist $\phi(1) = 1$.

Einige Werte von $\phi(n)$:

φ(n)	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9
0+	/	1	1	2	2	4	2	6	4	6
10+	4	10	4	12	6	8	8	16	6	18
20+	8	12	10	22	8	20	12	18	12	28
30+	8	30	16	20	16	24	12	36	18	24
40+	16	40	12	42	20	24	22	46	16	42
50+	20	32	24	52	18	40	24	36	28	58
60+	16	60	30	36	32	48	20	66	32	44
70+	24	70	24	72	36	40	36	60	24	78
80+	32	54	40	82	24	64	42	56	40	88
90+	24	72	44	60	46	72	32	96	42	60

Beispiel

$\phi(6) = 2 = |\{1, 5\}|$

Test:

$ggt(1, 6) = 1$✓

$ggt(2, 6) = 2$❌

$ggt(3, 6) = 3$❌

$ggt(4, 6) = 2$❌

$ggt(5, 6) = 1$✓

Vgl. https://de.wikipedia.org/wiki/Eulersche_Phi-Funktion

Eulers Theorem

besagt, dass für jedes a und n, die relativ prim sind:

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

Eine alternative Form ist:

$$a^{\phi(n)+1} \equiv a \pmod{n}$$

Miller-Rabin-Primzahltest

- Viele kryptografische Algorithmen erfordern eine oder mehrere sehr große Primzahlen nach dem Zufallsprinzip.
- Der Miller-Rabin-Primzahltest ist ein probabilistischer Primzahltest, der schnell und einfach ist.
- Hintergrund: Jede positive ungerade ganze Zahl $n \geq 3$ kann ausgedrückt werden als:

$n - 1 = 2^k q \qquad mit\ k > 0, q\ ungerade$

Miller-Rabin-Algorithmus

```
def MRTest(n, k): # n > 2, eine ungerade ganze Zahl,
                  #        die auf Primalität geprüft wird
                  # k,     die Anzahl der Testrunden

    let s > 0 and d odd > 0 such that n−1 = pow(2,s)*d
    repeat k times:
        a ← random(2, n−2)
        x ← pow(a,d) mod n
        repeat s times:
            y ← sqr(x) mod n
            if y = 1 and x ≠ 1 and x ≠ n−1 then return “composite”
            x ← y
        if y ≠ 1 then return “composite”
    return “probably prime”
```

Deterministische Primzahltests

- Vor 2002 gab es keine bekannte Methode, um für sehr große Zahlen effizient zu beweisen, dass diese Primzahlen sind.
- Alle verwendeten Algorithmen lieferten ein probabilistisches Ergebnis.
- Im Jahr 2002 entwickelten Agrawal, Kayal und Saxena einen Algorithmus, der „effizient“ bestimmt, ob eine gegebene große Zahl eine Primzahl ist:
  - Auch bekannt als AKS-Algorithmus.
  - Er scheint nicht so effizient zu sein wie der Miller-Rabin-Algorithmus.

Chinesischer Restsatz

(🇺🇸 *Chinese Remainder Theorem (CRT)*)

- Wurde vermutlich von dem chinesischen Mathematiker Sun-Tsu um 100 n. Chr. entdeckt [1].
- Eines der nützlichsten Ergebnisse der Zahlentheorie.
- Es besagt, dass es möglich ist, ganze Zahlen in einem bestimmten Bereich aus ihren Residuen modulo einer Menge von paarweise relativ primen Moduli zu rekonstruieren.
- Kann auf verschiedene Weise formuliert werden.

Bemerkung

Bietet eine Möglichkeit, (potenziell sehr große) Zahlen $\bmod M$ in Form von Tupeln kleinerer Zahlen zu manipulieren.

- Dies kann nützlich sein, wenn M 150 Ziffern oder mehr hat.
- Es ist jedoch notwendig, die Faktorisierung von M im Voraus zu kennen.

Bei RSA rechnen wir mit Zahlen mit weit über 300 Ziffern.

Von der Menge der paarweise relativ primen Moduli interessieren wir uns aber „nur“ für ein paar im Folgenden.

[1] Die Quellenlage bgzl. des genauen Datums ist unsicher und variiert teilweise um bis zu ca. 200 Jahre.

Chinesischer Restsatz - Beispiel in Z_{10}

Nehmen wir an, dass die (*relativ prim/koprimalen*) Faktoren einer Zahl x:

$m_1 = 2$ und $m_2 = 5$ sind.

Weiterhin seien die bekannten Reste der Division von x durch m_1 bzw. m_2: $a_1 = r_{m_1} = 0$ und $a_2 = r_{m_2} = 3$ sind.

D. h. $x \bmod 2 = 0$ und $x \bmod 5 = 3$; bzw. $x \equiv 0 \pmod{2}$ und $x \equiv 3 \pmod{5}$.

Da $x \bmod 2 = 0$ ist muss x eine gerade Zahl sein; außerdem ist $x \bmod 5 = 3$.

Die eindeutige Lösung in Z_{10} ist: 8.

Berechnung einer Lösung in Z:

$5 \times x_1 \equiv 1 \pmod{2}$	$x_1 = 1$	$x = a_1 \times m_2 \times x_1 + a_2 \times m_1 \times x_2$
$2 \times x_2 \equiv 1 \pmod{5}$	$x_2 = 3$	$x = 0 \times 5 \times 1 + 3 \times 2 \times 3 = 18$

Man könnte auch folgendes Problem versuchen zu lösen: Wir haben x Schokoladentafeln. Wenn wir diese fair auf zwei Personen verteilen, dann haben wir keinen Rest. Wenn wir diese jedoch auf 5 Personen aufteilen, dann haben wir 3 Tafeln übrig. Wieviele Schokoladentafeln haben wir?

(Zur Erinnerung: zwei Zahlen x und y sind relativ prim, wenn ihr größter gemeinsamer Teiler 1 ist.)

Chinesische Restsatz - Zusammenfassung

> Der chinesische Restsatz wird häufig für Berechnungen mit großen ganzen Zahlen verwendet, da er es ermöglicht, eine Berechnung, für die man eine Grenze für die Größe des Ergebnisses kennt, durch mehrere ähnliche Berechnungen mit kleinen ganzen Zahlen zu ersetzen.
>
> Das CRT findet in der Public-Key-Kryptographie Einsatz.

Übung

1. Berechne $5^9 \, mod \, 7$ ohne die Zuhilfenahme eines Taschenrechners.
2. Welche Zahlen sind relativ prim zu 21?
3. Berechne $ggt(1037, 768)$ mit Hilfe des Euklidischen Algorithmus.
4. Berechne $ggt(42, 16)$ mit Hilfe des erweiterten Euklidischen Algorithmus. D. h. berechnen Sie auch x und y!

Übung

1. Bestimme das Ergebnis von Euler's Totient Funktion ϕ für den Wert 37 ohne das Ergebnis nachzuschlagen.
2. Überzeugen Sie sich davon, dass der (kleine) Satz von Fermat gilt. Zum Beispiel für die Zahlen: $a = 9$ und $p = 7$.
3. Überzeugen Sie sich davon, dass der Satz von Euler gilt. Zum Beispiel für die Werte $a = 7$ und $n = 9$.
4. Führen Sie den Miller-Rabin Algorithmus für $n = 37$ aus.
5. In einer Tüte sind x Gummibärchen. Wenn Sie diese auf 4 Personen verteilen, dann haben Sie einen Rest von 2, verteilen Sie diese auf 7 Personen, dann haben Sie einen Rest von 3. Wie viele Gummibärchen sind in der Tüte? Wenden Sie den chinesischen Restsatz an.