Prof. Dr. Hartmut Ring
Universität Siegen

Diskrete Mathematik für Informatiker
Wintersemester 2010/2011

3.Elementare Zahlentheorie

Division mit Rest

Definition Teschl: 2.47

Für $a \in \mathbb{Z}$ und $m \in \mathbb{N}$ sind
das ***ganzzahlige Divisionsergebnis*** $q = a \text{ div } m$
und der ***Rest*** $r = a \bmod m$
(eindeutig!) definiert durch

$$0 \le r < m$$

und $a = q\,m + r$.

mathGUIde

a // m
a % m

Teilbarkeit

Definition Teschl: 2.39

Für zwei ganze Zahlen a und b
sagt man „a ***teilt*** b“ (geschrieben: $a \mid b$)
oder „b *ist* **teilbar** *durch* a“,
falls es eine Zahl $c \in \mathbb{Z}$ („***Teiler***“) gibt
mit $ac = b$

Für alle von Null verschiedenen Zahlen $a \in \mathbb{Z}$ gilt

$a \mid a$	$-a \mid a$
$1 \mid a$	$-1 \mid a$

Triviale Teiler

Definition Teschl: 2.39

Der ***größte gemeinsame Teiler*** von zwei natürlichen Zahlen a und b ist
$\text{ggT}(a, b) = \max \{n \in \mathbb{N} : n \mid a \wedge n \mid b\}$

a und b heißen ***teilerfremd***,
falls $\text{ggT}(a, b) = 1$

Englisch: *greatest common divisor* (gcd)

mathGUIde

```
gcd(a,b)
```

In der Zahlentheorie schreibt man (a,b) statt ggT(a,b)

Der euklidische Algorithmus

Satz Teschl: 3.33

Der folgende Algorithmus berechnet den größten gemeinsamen Teiler von zwei nicht negativen Zahlen a und b:

```
def gcd(a, b):
    if b == 0:
        return a
    else:
        return gcd(b, a % b)
```

ggT (a, 0) = a

Korrektheit des Algorithmus:

b > 0 ⇒ a und b haben die gleichen gemeinsamen Teiler wie a - b und b
⇒ ggT (a,b) = ggT (a-b, b)
= ggT (a-2b, b) = ...
= ggT (a mod b, b)
= ggT (b, a mod b)

Abbruch des Algorithmus:

Ist $a < b$, so werden im ersten rekursiven Aufruf die Parameter vertauscht. In jedem Fall ist nun $a \geq b$.

Wegen $0 \leq a \bmod b < b$ ist für jeden rekursiven Aufruf $a > b$ mit monoton fallendem $b \geq 0$.

Der Algorithmus bricht also nach weniger als b Schritten ab.

Iterative Variante

```
def gcd(a, b):
    while b > 0:
        a, b = b, a % b
    return a
```

Schritte im euklidischen Algorithmus

Rekursionsschritte für $a > b$

a \ b	0	1	2	3	4	5	6	7	8
1	0								
2	0	1							
3	0	1	2						
4	0	1	1	2					
5	0	1	2	3	2				
6	0	1	1	1	2	2			
7	0	1	2	2	3	3	2		
8	0	1	1	3	1	4	2	2	
9	0	1	2	1	2	3	2	3	2

kleinste Paare:

Schritte	a	b
0	1	0
1	2	1
2	3	2
3	5	3
4	8	5

Vermutung:
$a, b \in \mathbb{N}_0$, $a > b$
benötigt n Schritte
$\Rightarrow a \geq f_{n+2}$, $b \geq f_{n+1}$
(Fibonacci)

Schritte im euklidischen Algorithmus

a \ b	0	1	2	3
1	0			
2	0	1		
3	0	1	2	
4	0	1	1	2
5	0	1	2	3

Schritte	a	b
0	1	0
1	2	1
2	3	2
3	5	3
4	8	5

Vermutung:
a, $b \in \mathbb{N}_0$, a > b benötigt n Schritte
$\Rightarrow a \geq f_{n+2}$, $b \geq f_{n+1}$ (Fibonacci)

Beweis:

Induktionsanfang (n=1):
$\Rightarrow a \geq 2 = f_{1+2}$, $b \geq 1 = f_{1+1}$

Induktionsschritt:
a, b benötigt n+1 Schritte
1. Schritt: a mod b = r (a = qb+r)
b,r benötigt n Schritte
$\Rightarrow b \geq f_{n+2}$, $r \geq f_{n+1}$
$\Rightarrow a \geq qb+r \geq b+r \geq f_{n+3}$

Also: Anzahl der Rekursionsschritte
für ggT (a, b) im euklidischen Algorithmus:

$O(\log a)$

Erweiterter euklidischer Algorithmus

Satz Teschl: 3.35

Für beliebige a und $b \in \mathbb{N}$ gibt es x und $y \in \mathbb{Z}$ mit $xa + yb = \text{ggT}(a, b)$

Beispiel: ggT(456, 372) = 12

Einfach	Erweitert
456 = 1·372 + 84	84 = 456 - 1·372
372 = 4· 84 + 36	36 = 372 - 4·84 = 372 - 4·(456 - 1·372) = 5·372 - 4·456
84 = 2· 36 + 12	12 = 84 - 2·36 = (456 - 1·372) - 2·(5·372 - 4·456) = 9·456 - 11·372
36 = 3· 12 + 0	

Erweiterter euklidischer Algorithmus

456 ⟶ (1, 0)

372 ⟶ (0, 1)

456 = 1·372 + 84	84 ⟶ 1·(1, 0) – 1·(0,1) = (1, -1)
372 = 4· 84 + 36	36 ⟶ 1·(0, 1) – 4·(1,-1) = (-4, 5)
84 = 2· 36 + 12	12 ⟶ 1·(1, -1) – 2·(-4, 5) = (9, -11)
36 = 3· 12 + 0	

```
def gcdX(a, b):                                   z.B. 456, 372
    a0, b0 = a, b
    m = {a:Vector([1,0]), b:Vector([0,1])}
    while b > 0:
        q, r = a // b, a % b                      1, 84
        assert a == q*b + r
        assert r == a - q*b
        m[r] = m[a] - q*m[b]                      84 = (1*456+0*372) - 1*(0*456+1*372)
        a, b = b, r                               372, 84
    x, y = m[a][0], m[a][1]
    assert a == x * a0 + y * b0
    return a, x, y
```

Erweiterter euklidischer Algorithmus

Satz Teschl: 3.35

Für beliebige a und $b \in \mathbb{N}$ gibt es x und $y \in \mathbb{Z}$ mit $xa + yb = \text{ggT}(a, b)$

Beweis durch Verifikation des Algorithmus:

```
def gcdExt(a,b):
    if b == 0:
        (d, x, y) = (a, 1, 0)
    else:
        (q, r) = divmod(a, b)
        (d, z, x) = gcdExt(b, r)
        y = z - x * q
    assert d == gcd(a,b)
    assert d == x*a + y*b
    return (d, x, y)
```

ggT (a,b) = a = 1·a + 0·b

r = a – b·q

nach Induktionsvoraussetzung:
d = z·b + x·r
= z·b + x·(a – b·q)
= x·a + b·(z – x·q)

d = x·a + b·y

mathGUIde

```
gcdExt(a,b)
```


Primzahlen

Definition Teschl: 2.41

Eine natürliche Zahl $p > 1$ heißt ***Primzahl***, falls sie nur die positiven trivialen Teiler 1 und p besitzt.

mathGUIde

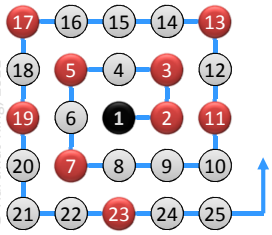
```
isPrime(n)
prime(i)
nextPrime(n)
```

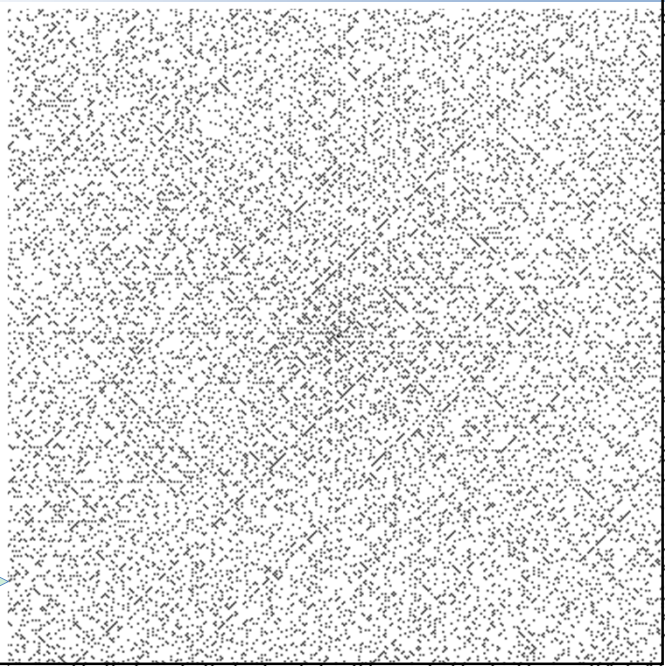
Satz Teschl: 2.43

Jede natürliche Zahl ist Produkt endlich vieler Primzahlen.

Beweis:
1 ist leeres Produkt.
Annahme: Der Satz ist falsch.
Dann gibt es ein kleinstes a mit:
a ist nicht Produkt endlich vieler Primzahlen.
Also ist a keine Primzahl (und $a > 1$).
a lässt sich also schreiben als $a = b\,c$ mit $b, c < a$.
Das ist ein Widerspruch zur Annahme!

Primzahlen



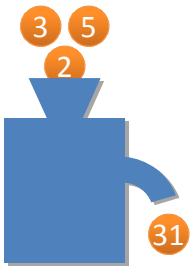


Die Primzahlen bis $399^2 = 159201$

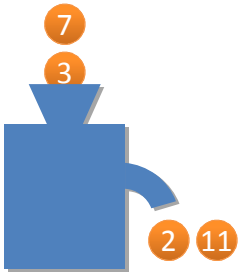
Idee: Stanislaw Ulam (1909-1986)

Primzahlen

Eine Primzahlmaschine aus der Antike



$2 \cdot 3 \cdot 5 + 1 = 31$



$3 \cdot 7 + 1 = 22 = 2 \cdot 11$

Satz **Euklid** (um 300 v. Chr.) Teschl: 2.44

Es gibt unendlich viele Primzahlen.

Primzahlen

Satz -

Ist p Primzahl, $a, b \in \mathbb{N}$ und $p \mid ab$,
so folgt: $p \mid a$ oder $p \mid b$

Beweis: Gilt $p \mid a$, so ist nichts zu beweisen.
Andernfalls ist ggT(p,a) = 1.
Also gibt es $x, y \in \mathbf{Z}$ mit $xp + ya = 1$.
Multiplikation mit b ergibt $bxp + bya = b$.
Wegen $p \mid p$ und $p \mid ab$ folgt $p \mid b$.

Satz -

Ist p Primzahl, $a_1, a_2, \ldots, a_n \in \mathbb{N}$ und $p \mid a_1, a_2, \ldots, a_n$,
so folgt: Es gibt ein $i \in \{1,\ldots,n\}$ mit $p \mid a_i$

Beweis: Verallgemeinerung der obigen Aussage mit Vollständiger Induktion!

Primzahlen

Satz -

$p_1, p_2, \ldots p_r$ und $q_1, q_2, \ldots q_s$ seien Primzahlen
mit $p_1 p_2 \ldots p_r = q_1 q_2 \ldots q_s$.
Dann ist $r = s$ und $(p_1, p_2, \ldots p_r)$ eine Permutation von $(q_1, q_2, \ldots q_s)$.

Beweis durch starke vollständige Induktion über $n = p_1 p_2 \ldots p_r = q_1 q_2 \ldots q_s$:

Für $n = 1$ sind beide Produkte leer und damit die Behauptung richtig.

Die Behauptung gelte nun für 1, 2, … , n.

$n+1 = p_1 p_2 \ldots p_r = q_1 q_2 \ldots q_s$.
Aus $p_1 \mid p_1 p_2 \ldots p_r$ folgt $= p_1 \mid q_1 q_2 \ldots q_s$. Also gibt es ein i mit $p_1 \mid q_i$.

Durch geeignete Umnummerierung wird daraus $p_1 \mid q_1$,
also $p_1 = q_1$ (da p_1 und q_1 Primzahlen sind).

Nun sei $n' = p_2 \ldots p_r = q_2 \ldots q_s$.
Wegen $n' < n+1$ folgt mit der Induktionsvoraussetzung
$r = s$ und $(p_2, \ldots p_r)$ ist eine Permutation von $(q_2, \ldots q_s)$.
Daraus folgt die Behauptung für $n+1$.

Primzahlen

Satz -

$p_1, p_2, \ldots p_r$ und $q_1, q_2, \ldots q_s$ seien Primzahlen
mit $p_1 p_2 \ldots p_r = q_1 q_2 \ldots q_s$.
Dann ist $r = s$ und $(p_1, p_2, \ldots p_r)$ eine Permutation von $(q_1, q_2, \ldots q_s)$.

Satz **Fundamentalsatz der Zahlentheorie**: Teschl: 2.43

Für jede natürliche Zahl n gibt es eine *eindeutige* ***Primfaktorenzerlegung***:

$$n = \prod_{i=1}^{m(n)} p_i^{e_i(n)}$$

mathGUIde

```
factor(n)
factors(n)
allFactors(n)
```


Experimente mit Primzahlen (1)

3,5 5,7 11,13 17,19 29,31

mathGUIde

```
[i for i in fromTo(1, 100) if isPrime(i) and isPrime(i+2)]
```

Gibt es unendlich viele ***Primzahlzwillinge***?

Experimente mit Primzahlen (2)

4 = 2+2
6 = 3+3
8 = 3+5
10 = 3+7 = 5+5
12 = 5+7
14 = 3+11 = 7+7
16 = 3+13 = 5+11
18 = 5+13 = 7+11
20 = 3+17 = 7+13
22 = 3+19 = 5+17 = 11+11

Goldbachsche Vermutung:
(1742 Brief an Euler)

Jede gerade Zahl ab 4 ist Summe zweier Primzahlen.

Bisher unbewiesen!

mathGUIde

```
for n in fromTo(6, 100, 2):
    print(n, end=" ")
    for i in fromTo(3, n//2, 2):
        if isPrime(i) and isPrime(n-i):
            print("= {0} + {1}".format(i, n-i), end=" ")
    print()
```


Experimente mit Primzahlen (3)

Fermatsche Primzahlen
Pierre de Fermat (1601-1665)

$$F_n = 2^{2^n} + 1$$

?

$F_1 = 5$

$F_2 = 17$

$F_3 = 257$

$F_4 = 65537$

$F_5 = 4294967297 = 641 \cdot 6700417$

F_5 bis F_{30} sind keine Primzahlen.
Bisher ist keine weitere Fermatsche Primzahl bekannt.

mathGUIde

```
[2^2^n+1  for n in fromTo(1, 4)]
```

Experimente mit Primzahlen (4)

Mersenne-Primzahlen
Marin Mersenne (1588-1648)

$$2^p - 1$$

(*p* Primzahl)

?

$2^2-1 = 3$

$2^3-1 = 7$

$2^5-1 = 31$

$2^7-1 = 127$

$2^{11}-1 = 2047 = 23 \cdot 89$

mathGUIde

```
[2^prime(i)-1 for i in fromTo(1, 4)]
```


Mersenne-Primzahlen

Nr.	Tag	Zahl	Ziffern
1		2^2-1	1
2		2^3-1	1
3		2^5-1	2
4		2^7-1	3
5	1456	$2^{13}-1$	4
...			
8	1772	$2^{31}-1$	10
...			
18	1957	$2^{3.217}-1$	969
...			
28	1982	$2^{86.243}-1$	25.962

Nr.	Tag	Zahl	Ziffern	
37	27. 01. 1998	$2^{3.021.377}-1$	909.526	
38	01. 06. 1999	$2^{6.972.593}-1$	2.098.960	
39	05. 11. 2001	$2^{13.466.917}-1$	4.053.946	
40	17. 11. 2003	$2^{20.996.011}-1$	6.320.430	1
41 (?)	15. 05. 2004	$2^{24.036.583}-1$	7.235.733	
42 (?)	18. 02. 2005	$2^{25.964.951}-1$	7.816.230	
43 (?)	15. 12. 2005	$2^{30.402.457}-1$	9.152.052	
44 (?)	04. 09. 2006	$2^{32.582.657}-1$	9.808.358	
45 (?)	06. 09. 2008	$2^{37.156.667}-1$	11.185.272	
46 (?)	12. 04. 2009	$2^{42.643.801}-1$	12.837.064	
47 (?)	23. 08. 2008	$2^{43.112.609}-1$	12.987.189	2

1 Möglicherweise gibt es zwischen Zeile 40 und Zeile 47(?) noch weitere Mersenne-Primzahlen!

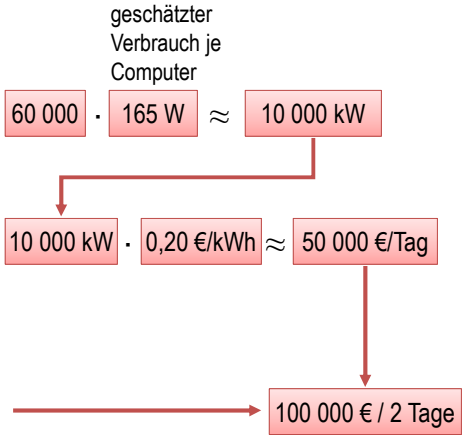
2 Derzeit größte bekannte Primzahl

Primzahl-Ökologie

Das GIMPS-Projekt (Great Internet Mersenne Prime Search) vermeldet auf seiner Homepage die Entdeckung der 44. mersenneschen Primzahl. **Insgesamt über 60.000 Computer von Freiwilligen** beteiligen sich im Rahmen dieses Projekts zurzeit an der Suche nach immer größeren Primzahlen und verbraten dazu gemeinsam eine Rechenleistung von über 20 Teraflop/s. Einer davon ist am 4. September fündig geworden.

Sollte die Zahl mehr als 10 Millionen Stellen haben, winken **100.000 US-Dollar Preisgeld**, das die Electronic Frontier Foundation (EFF) dafür ausgelobt hat. Die ebenfalls vom GIMPS-Projekt gefundene 43. mersennesche Primzahl $2^{30.402.457}-1$ liegt mit 9.152.052 Dezimalstellen noch knapp unter dieser Grenze.

(c't news, 06.09.2006)



M_{44} in der Presse

Zwei US-Forscher haben die bislang größte Primzahl entdeckt. Sie hat ausgeschrieben 9 808 358 Stellen.
Curtis Cooper und Steven Boone ... verpassen mit ihrem Rekord **nur knapp** das von der Electronic Frontier Foundation ausgelobte Preisgeld von 100.000 Dollar für die erste Primzahl mit mehr als **zehn Millionen Stellen**.

Siegener Zeitung, 21. 10. 2006

Wer wird Dezimillionär?

Wieviel Prozent der „10-Millionen-Grenze“ haben die beiden erreicht (auf 8 Stellen nach dem Komma gerundet) ?

A 98,08358 %

B 9,808358 %

C 0,9808358 %

D 0,00000000 %

Experimente mit Primzahlen (5)

mathGUIde

```
p = nextPrime(10^5)
q = nextPrime(10^5 + 10^4)
n = p * q

factor(n)
```

Vorschau: *Kryptographie*:
RSA-Algorithmus
(R. Rivest, A. Shamir, L. Adleman 1978)

$m \longrightarrow c = m^e \bmod n \longrightarrow m = c^d \bmod n$

um d aus e berechnen zu können, muss man p und q kennen!

The RSA Challenge Numbers

http://www.rsasecurity.com/rsalabs/node.asp?id=2091

F. Bahr, M. Boehm, J. Franke, T. Kleinjung

BSI (Bundesamt für Sicherheit in der Informationstechnik)

5 Monate Rechenzeit auf einem Opteron-Cluster mit 80 2,2-GHz-Prozessoren

3107418240490043721350750035888567930037346022842727545720161948823206440518081504556346829671723286782437916272838033415471073108501919548529007337724822783525742386454014691736602477652346609
=
1634733645809253848443133883865090859841783670033092312181110852389333100104508151212118167511579
*
1900871281664822113126851573935413975471896789968515493666638539088027103802104498957191261465571

Zahl	Dezimal-stellen	Preis	faktorisiert
RSA-100	100		April 1991
RSA-110	110		April 1992
RSA-120	120		Juni 1993
RSA-129	129	100$	April 1994
RSA-130	130		April 1996
RSA-140	140		Feb. 1999
RSA-150	150		April 2004
RSA-155	155		Aug. 1999
RSA-160	160		April 2003
RSA-200	200		9. Mai 2005
RSA-576	174	10000$	Dez. 2003
RSA-640	193	20000$	2. Nov. 2005
RSA-704	212	30000$	
RSA-768	232	50000$	
RSA-896	270	75000$	
RSA-1024	309	100000$	
RSA-1536	463	150000$	
RSA-2048	617	200000$	

Rechnen mit Kongruenzen

Definition Teschl: 3.1

$a, b \in \mathbb{Z}$ heißen **kongruent** modulo $m \in \mathbb{N}$,
geschrieben: $a \equiv b \pmod m$,
falls $a \bmod m = b \bmod m$

Satz Teschl: 3.4

1. $a \equiv a'$ und $b \equiv b' \Rightarrow a+b \equiv a'+b' \pmod m$
2. $a \equiv a'$ und $b \equiv b' \Rightarrow ab \equiv a'b' \pmod m$
3. $a \equiv b \Rightarrow a^n \equiv b^n \pmod m$
4. $ad \equiv bd \Leftrightarrow a \equiv b \pmod m$, falls d und m teilerfremd

Beweis zu ❹:

$\Leftarrow$ klar!

$\Rightarrow$ Nach erweitertem Euklid gibt es x und y mit $xd + ym = 1$
$\Rightarrow xd \equiv 1 \pmod m$

$ad \equiv bd \Rightarrow axd \equiv bxd \Rightarrow a \equiv b \pmod m$

Chinesischer Restsatz (Vorbereitung)

$x \bmod 3 = 0$
$x \bmod 6 = 1$

keine Lösung!

$x \bmod 2 = 1$
$x \bmod 3 = 2$
$x \in \{0, 1, \ldots, 5\}$

eine Lösung: $x = 5$

Chinesischer Restsatz (Vorbereitung)

$x \bmod m_1 = a_1$
$x \bmod m_2 = a_2$

m_1 und m_2 teilerfremd

Für zwei Lösungen x und x' gilt:
$x \bmod m_1 = x' \bmod m_1 = a_1$
$x \bmod m_2 = x' \bmod m_2 = a_2$

→ $|x - x'|$ Vielfaches von m_1
$|x - x'|$ Vielfaches von m_2

→ $|x - x'|$ Vielfaches von $m_1 m_2$

→ höchstens eine Lösung in $\{0, 1, \ldots, m_1 m_2 - 1\}$

Die $m_1 m_2$ Paare
$[0 \bmod m_1, 0 \bmod m_2]$,
$[1 \bmod m_1, 1 \bmod m_2]$, ...
$[m_1 m_2 - 1 \bmod m_1, m_1 m_2 - 1 \bmod a_2]$
sind alle verschieden,
enthalten also jede Kombination genau einmal.

sonst gäbe es zwei Lösungen

Chinesischer Restsatz Sun-Tsu (1. Jhd.)

Satz Teschl: 3.43

$m_1, m_2, \ldots, m_r$ seien paarweise teilerfremde ganze Zahlen,
$n := m_1 \cdot m_2 \cdot \ldots \cdot m_r$
Dann hat das Gleichungssystem

$x \bmod m_1 = a_1 \quad (0 \le a_1 < m_1)$
...
$x \bmod m_r = a_r \quad (0 \le a_r < m_r)$

genau eine Lösung $x \in \{0, 1, \ldots, n-1\}$

x mod 5 = 4

x mod 7 = 3

x ∈ {0, 1, ..., 34}

mathGUIde

```
chinese (Mod(4,5), Mod(3,7))
```

Chinesischer Restsatz: Beispiel

Gesucht	$n \in \{0, 1, ..., 76\}$	$n \in \{0, 1, ..., \mathbf{p} \cdot \mathbf{q}\text{-}1\}$
mit	$n \bmod \mathbf{7} = 4$	$n \bmod \mathbf{p} = a$
	$n \bmod \mathbf{11} = 5$	$n \bmod \mathbf{q} = b$

Erweiterter Euklid $(-3) \cdot \mathbf{7} + 2 \cdot \mathbf{11} = 1$ | $x \cdot \mathbf{p} + y \cdot \mathbf{q} = 1$

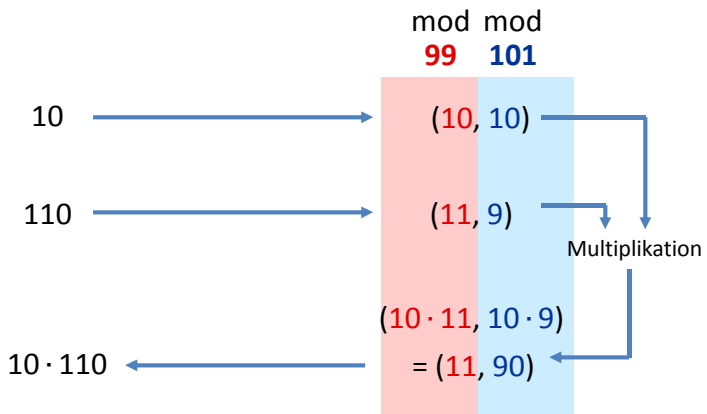
	mod 7	mod 11		mod p	mod q
$(-3) \cdot \mathbf{7}$	0	1	$x \cdot \mathbf{p}$	0	1
$2 \cdot \mathbf{11}$	1	0	$y \cdot \mathbf{q}$	1	0
$5 \cdot (-3) \cdot \mathbf{7}$	0	5	$b \cdot x \cdot \mathbf{p}$	0	b
$4 \cdot 2 \cdot \mathbf{11}$	4	0	$a \cdot y \cdot \mathbf{q}$	a	0
$5 \cdot (-3) \cdot \mathbf{7} + 4 \cdot 2 \cdot \mathbf{11}$	4	5	$b \cdot x \cdot \mathbf{p} + a \cdot y \cdot \mathbf{q}$	a	b
-17	4	5			

$n = (-17) \bmod 77 = 60$ | $n = (b \cdot x \cdot \mathbf{p} + a \cdot y \cdot \mathbf{q}) \bmod \mathbf{p} \cdot \mathbf{q}$

Modulare Arithmetik

$p = 99$ $q = 101$

99 und 101 teilerfremd.
Jede Zahl von 0 bis $99 \cdot 101\text{-}1 = 9998$
ist eindeutig als Restepaar darstellbar.



Vollständige Restsysteme

$ad \equiv bd \Leftrightarrow a \equiv b \pmod m$, falls d und m teilerfremd — Satz 3.4

n	0	1	2	3	4	5	6
$5n$	0	5	10	15	20	25	30
$5n \bmod 7$	0	5	3	1	6	4	2

Vollständige Restsysteme modulo 7

$\{0,1,...,6\} = \{5 \cdot 0 \bmod 7, 5 \cdot 1 \bmod 7, ..., 5 \cdot 6 \bmod 7\}$

$\{0,1,...,6\} \equiv \{5 \cdot 0, 5 \cdot 1, ..., 5 \cdot 6\} \pmod 7$

$\{0,1,...,6\} \equiv 5 \cdot \{0, 1, ..., 6\} \pmod 7$

vereinfachte Schreibweisen

Vollständige Restsysteme

Satz -

Wenn ggT(d, p) = 1,
dann ist $\{d \cdot 0, d \cdot 1, ..., d \cdot (p\text{-}1)\}$ ein vollständiges Restsystem modulo p.

Beweis:

$a, b \in \{0,..., p\text{-}1\}$, $a \neq b$, ggT(d, p) = 1

$\Rightarrow$ nicht $a \equiv b \pmod p$

$\Rightarrow$ nicht $ad \equiv bd \pmod p$ — Satz 3.4

also: $\{d \cdot 0, d \cdot 1, ..., d \cdot (p\text{-}1)\}$ paarweise inkongruent

Die eulersche Phi-Funktion

Definition Teschl: S. 79/85

Für alle $n \in \mathbb{N}$ wird definiert:

$$\mathbb{Z}_n = \{0, 1, 2, \ldots, n\text{-}1\}$$

$$\mathbb{Z}^*_n = \{a \in \mathbb{Z}_n \mid \text{ggT}(a,n) = 1\}$$

n	$\mathbb{Z}^*_n$	n	$\mathbb{Z}^*_n$
1	$\emptyset$	7	{1,2,3,4,5,6}
2	{1}	8	{1,3,5,7}
3	{1,2}	9	{1,2,4,5,7,8}
4	{1,3}	10	{1,3,7,9}
5	{1,2,3,4}	11	{1,2,3,...,10}
6	{1,5}	12	{1,5,7,11}

p Primzahl $\Rightarrow |\mathbb{Z}^*_p| = p - 1$

Die eulersche Phi-Funktion

Definition Teschl: S. 102

Für alle natürlichen Zahlen ist die eulersche Phi-Funktion definiert als

$$\varphi(n) = |\{i \in \{1,2,\ldots, n\} : \text{ggT}(i,n) = 1\}|$$

n		$\varphi(n)$
1	{1}	1
2	{1}	1
3	{1,2}	2
4	{1,3}	2
5	{1,2,3,4}	4
6	{1,5}	2
7	{1,...,6}	6
8	{1,3,5,7}	4
9	{1,2,4,5,7,8}	6
10	{1,3,7,9}	4

Für $n > 1$ gilt $\varphi(n) = |\mathbb{Z}^*_n|$

mathGUIde (ineffiziente Implementierung!)

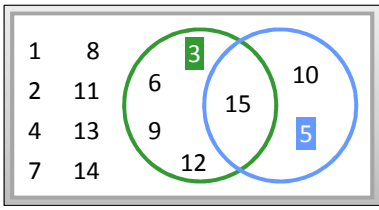
```
def phi(n):
    return len([i for i in fromTo(1, n)
                if gcd(i,n) == 1])

[phi(i) for i in fromTo(1, 10)]
```

p Primzahl $\Rightarrow \varphi(p) = p - 1$

Die Eulersche Phi-Funktion

$$\varphi(n) = |\{i \in \{1,2,\ldots, n\} : ggT(i,n) = 1\}|$$

p Primzahl $\Rightarrow \varphi(p) = p - 1$



$\varphi(3 \cdot 5)$
$= 15 - 5 - 3 + 1$
$= (3-1)(5-1)$

$n = pq$ (Primzahlen) $\Rightarrow \varphi(n) = (p-1)(q-1)$

$$= n \cdot \frac{p-1}{p} \cdot \frac{q-1}{q}$$

mathGUIde-Implementierung:

```
def eulerPhi(n):
    for p in factors(n):
        n = n * (p-1) // p
    return n
```


Kleiner Satz von Fermat

Satz **Kleiner Satz von Fermat** Teschl: 3.42

Wenn...	dann gilt:
❶ $n \in \mathbb{N}$	
❷ p ist Primzahl	$n^{p-1} \equiv 1 \pmod p$
❸ $ggT(n, p) = 1$	

Beweis:

Wegen ❸ ist $\{n \cdot 0, n \cdot 1, \ldots, n \cdot (p-1)\}$
ein vollständiges Restsystem modulo p (vgl. Folie 30).

$\Rightarrow \{n, 2n, \ldots, (p-1) \cdot n\} \equiv \{1, 2, \ldots, p-1\} \pmod p$

$\Rightarrow n \cdot 2n \cdot \ldots \cdot (p-1) \cdot n \equiv (p-1)! \pmod p$

$\Rightarrow (p-1)! \cdot n^{p-1} \equiv (p-1)! \pmod p$

$\Rightarrow n^{p-1} \equiv 1 \pmod p$ (Satz 3.4)

$(p-1)!$ und p sind teilerfremd!

Kleiner Satz von Fermat: Variante

Satz Kleiner Satz von Fermat

Wenn...	dann gilt:
❶ $n \in \mathbb{N}$	
❷ p ist Primzahl	$n^{p-1} \equiv 1 \pmod{p}$
❸ ggT$(n, p) = 1$	

Variante

Wenn...	dann gilt:
❶ $n \in \mathbb{N}$	
❷ p ist Primzahl,	$n^p \equiv n \pmod{p}$

Beweis:

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

Fall (b): ggT $(n, p) \neq 1 \Rightarrow p \mid n \Rightarrow n \equiv 0 \Rightarrow n^p \equiv 0 \equiv n \pmod{p}$

Verallgemeinerung: Satz von Euler/Fermat

Satz n.n (Kleiner Satz von Fermat)

Wenn...	dann gilt:
❶ $n \in \mathbb{N}$	
❷ p ist Primzahl	$n^{p-1} \equiv 1 \pmod{p}$
❸ ggT$(n, p) = 1$	

Satz von Euler/Fermat

Wenn...	dann gilt:
❶ $n \in \mathbb{N}$	
❷ $a \in \mathbb{N}$	$n^{\varphi(a)} \equiv 1 \pmod{a}$
❸ ggT$(n, a) = 1$	

Satz von Euler/Fermat

Satz von Euler/Fermat

Wenn...	dann gilt:
❶ $n \in \mathbb{N}$	$n^{\varphi(a)} \equiv 1 \pmod a$
❷ $a \in \mathbb{N}$	
❸ ggT$(n, a) = 1$	

Hilfsbehauptung:
Für je zwei verschiedene Elemente $x, y \in M$ ist $xn \not\equiv yn \pmod a$.

Beweis:
Da x, y und n teilerfremd zu a sind, sind auch xn und yn teilerfremd zu a.
Also $xn \bmod a \in M$
und $yn \bmod a \in M$.
Aus $xn \equiv yn$ würde $x \equiv y \pmod a$ folgen, weil ggT$(a,n) = 1$ (nach Satz 3.4 ❹).
Also sind xn und yn nicht kongruent.

Beweisidee am Beispiel: $n = 5$, $a = 8$

Menge der zu a teilerfremden Zahlen $< a$:

$M = \mathbb{Z}^*_8 = \{1, 3, 5, 7\}$, $|M| = \varphi(a) = 4$

$\{1n, 3n, 5n, 7n\} = \{5 \cdot 1, 5 \cdot 3, 5 \cdot 5, 5 \cdot 7\}$
$= \{5, 15, 25, 35\} \equiv \{1, 3, 5, 7\} \pmod a$

$\Rightarrow 1n \cdot 3n \cdot 5n \cdot 7n \equiv 1 \cdot 3 \cdot 5 \cdot 7 \pmod a$

$\Rightarrow 1 \cdot 3 \cdot 5 \cdot 7 \cdot n^{\varphi(a)} \equiv 1 \cdot 3 \cdot 5 \cdot 7 \pmod a$

$\Rightarrow n^{\varphi(a)} \equiv 1 \pmod a$ (nach Satz 3.4 ❹)

Umkehrung des Satzes von Fermat?

Fermat: p Primzahl $\Rightarrow n^p \equiv n \pmod p$
$\Rightarrow (n^p - n) \bmod p = 0$

	$p = 3$	$p = 4$	$p = 5$
$n = 2$	$(2^3 - 2) \bmod 3 = 0$	$(2^4 - 2) \bmod 4 = 2$	$(2^5 - 2) \bmod 5 = 0$
$n = 3$	$(3^3 - 3) \bmod 3 = 0$	$(3^4 - 3) \bmod 4 = 2$	$(3^5 - 3) \bmod 5 = 0$
$n = 4$	$(4^3 - 4) \bmod 3 = 0$	$(4^4 - 4) \bmod 4 = 0$	$(4^5 - 4) \bmod 5 = 0$

mathGUIde

```
n = 2
for p in fromTo(3, 1000):
    if not isPrime(p) and (n^p - n) % p == 0:
        print(p)
```

n = 561 ???

Erste *Carmichael-Zahl*

2163 Carmichael-Zahlen bis 25 000 000 000

Satz von Wilson

mathGUIde

```
for n in fromTo(2, 20):
    print([n, isPrime(n), (factorial(n-1)+1) % n])
```

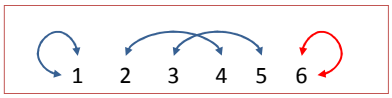
Satz von Wilson:
Für natürliche Zahlen $n > 1$ gilt:
n ist Primzahl $\Leftrightarrow$ $(n\text{-}1)! \equiv -1 \pmod{n}$

Beweisskizze: *n* Primzahl, z.B. 7:

Erweiterter Euklid: $x \cdot 3 + y \cdot 7 = 1$

(3, 7 teilerfremd) $5 \cdot 3 - 2 \cdot 7 = 1$

$\Rightarrow x \cdot 3 \equiv 1 \pmod{7}$



5 und 3 sind *multiplikative Inverse*. x zu sich selbst invers:

$x \cdot x \equiv 1 \Leftrightarrow x^2 - 1 \equiv 0 \Leftrightarrow (x+1)(x-1) \equiv 0 \Leftrightarrow x \equiv 1$ oder $x \equiv -1$

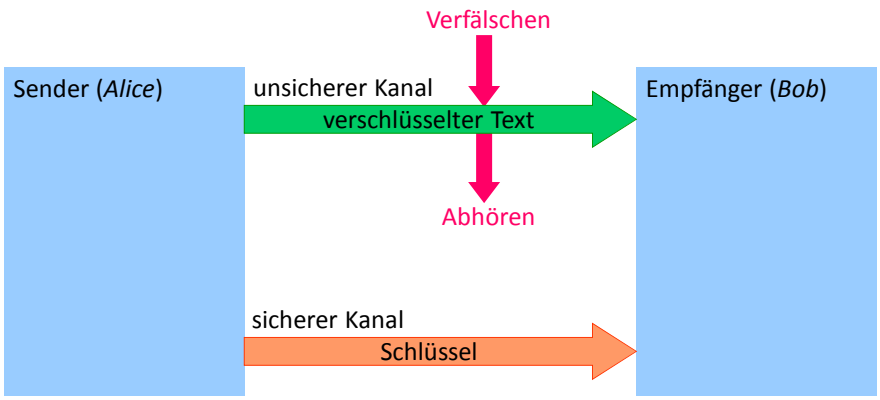
Rest: Paare mit Produkt 1

***n* keine Primzahl**:

$\Rightarrow$ in $\{1,2,\ldots,n-1\}$ kommen alle Primfaktoren von n vor

$\Rightarrow (n-1)!$ teilbar durch n. Also $(n-1)! \equiv 0 \pmod{n}$

Klassische Chiffrierung



Klassische Chiffrierung

A	0
B	1
C	2
D	3
E	4
F	5
G	6
H	7
I	8
J	9
K	10
L	11
M	12
N	13
O	14
P	15
Q	16
R	17
S	18
T	19
U	20
V	21
W	22
X	23
Y	24
Z	25

Caesar-Chiffre:
m[i] → (m[i]+d) mod 26

Extrem unsicher!

m GEHEIM
d=1
HFIFJN

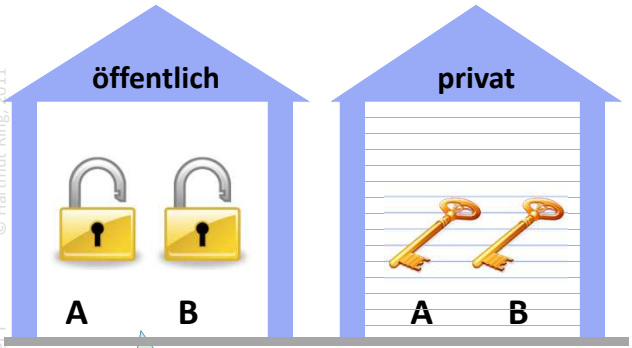
Vigenère-Chiffre:
m[i] →
(m[i]+k[i mod r]) mod 26

m ANANASPLANTAGE Nachricht
k BANANEBANANEBANANE Schlüssel (Länge r)
BNNNNWQLNNGEHE

Vernam-Chiffre:
m[i] →
(m[i]+k[i]) mod 26
Wie Vigenère, aber Schlüssel so lang wie Nachricht.
Sicher, falls der Schlüssel zufällig ist und nur einmal verwendet wird.

m ANANASPLANTAGE Nachricht
k BANANENDAMPFER Schlüssel
BNNNNWCOAZIFKV

Öffentliche Schlüssel

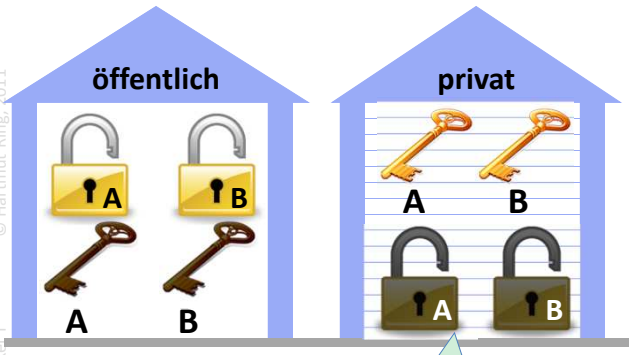


Schlösser können nach dem Zudrücken nur mit dem passenden Schlüssel geöffnet werden.

Nachricht von A an B

Sichere Kommunikation

Öffentliche Schlüssel



Einwegschlösser:
Können nur einmal (und nur mit dem passenden Schlüssel) geöffnet werden.

Nachricht von A an B

Sichere Kommunikation mit elektronischer Unterschrift

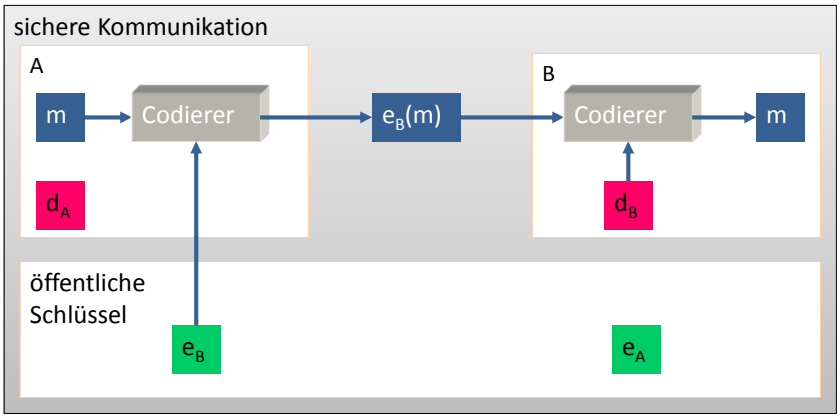
Öffentliche Schlüssel

Öffentlicher Schlüssel:
bijektive Abbildung e mit Umkehrabbildung $d = e^{-1}$

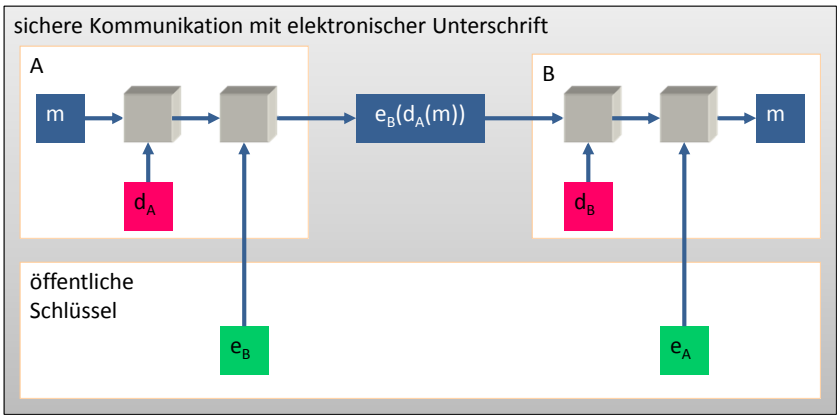
- Algorithmus für e wird veröffentlicht.
- Den Algorithmus für d kennt nur der Besitzer von e.
- Aus dem Algorithmus für e lässt sich kein effizienter Algorithmus für d ermitteln ("Einwegfunktion").

Beispiel: Name → Telefonnummer (mit Telefonbuch)

Öffentliche Schlüssel



Öffentliche Schlüssel



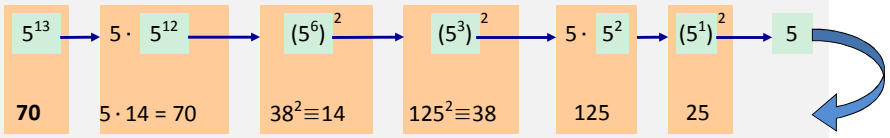
Exponentiations-Chiffren

$f: \{0, ..., n\text{-}1\} \to \{0, ..., n\text{-}1\}$
$f(a) = a^x \bmod n$

mathGUIde

```
expMod (5, 13, 143)
```

Beispiel: $5^{13} \bmod 143$




```
def expMod(a, x, n):
    if x == 1:
        return a % n
    elif x % 2 == 1:
        return (a * expMod(a, x-1, n)) % n
    else:
        return expMod(a, x // 2, n) ^ 2 % n
```

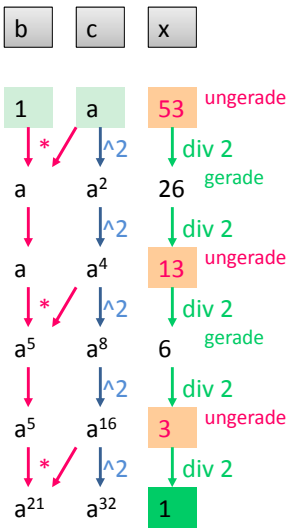

Exponentiations-Chiffren

Iterative Variante

```
def expMod (a, x, n):
  (b, c) = (1, a)
  while x > 1:
    if x % 2 == 1:
      b = (b*c) % n
    c = (c * c) % n
    x = x // 2
  return (b*c) % n
```

Syntax für TI 89/92
(Rekursion führt schnell zum Stack-Überlauf)

```
:expMod (a, x, n)
:Func
:Local b, c
:1 → b
:a → c
:While x > 1
:  If mod (x, 2) = 1 Then
:    mod (b*c, n) → b
:  EndIf
:  mod (c*c, n) → c
:  intDiv (x, 2) → x
:EndWhile
:Return mod (b*c, n)
:EndFunc
```



Invariante: $b \cdot c^x = a^{53}$

Das RSA-Verfahren

R. **R**ivest, A. **S**hamir, L. **A**dleman (1978)

Satz **RSA-Verfahren** Teschl: 3.42

Wenn...	dann gilt:
❶ $ed \equiv 1 \pmod{\varphi(n)}$	$(m^e)^d \equiv m \pmod{n}$
❷ $m \in \{0,1,\ldots,n-1\}$	
❸ $\text{ggT}(m,n) = 1$	

Zur Erinnerung:

Satz von Euler/Fermat

Wenn...
❶ $n \in \mathbb{N}$
❷ $a \in \mathbb{N}$
❸ $\text{ggT}(n, a) = 1$
dann gilt:
$n^{\varphi(a)} \equiv 1 \pmod{a}$

Beweis:

$ed \equiv 1 \pmod{\varphi(n)}$

$\Rightarrow \exists\, r > 0$ mit $ed = r\,\varphi(n) + 1$

$\Rightarrow m^{ed} = m^{r\varphi(n)+1}$

$= m\, m^{r\varphi(n)}$

$\equiv m \pmod{n}$

$m^{r\varphi(n)} = (m^{\varphi(n)})^r$
$\equiv 1 \pmod{n}$
nach Satz von Euler/Fermat

Das RSA-Verfahren

Schlüsselerzeugung

1	Wähle zwei Primzahlen p, q	p=11, q=13
2	n := pq, φ(n) = (p-1)(q-1)	n = 143, φ(n) = 120
3	d ∈ {1, ..., φ(n)} mit ggt (d, φ(n)) = 1	d = 37
4	Mit erweitertem euklidischen Algorithmus x und y bestimmen: xd + y φ(n) = 1 e := x mod φ(n) ⇒ ed ≡ 1 (mod φ(n))	x = 13, y = -4 e = 13 ed = 481 = 4 · 120+1
	Öffentlicher Schlüssel: (e,n)	(13,143)
	Privater Schlüssel: (d,n)	(37,143)

Kommunikation

		m = 5
●	Chiffrierung: $m \rightarrow c = m^e \bmod n$	$c = 5^{13} \bmod 143 = 70$
●	Dechiffrierung: $c \rightarrow m = c^d \bmod n$	$m = 70^{37} \bmod 143 = 5$

Das RSA-Verfahren

Die **Sicherheit des RSA-Verfahrens** beruht darauf, dass Produkte zweier großer Primzahlen nicht effizient faktorisiert werden können.

Am 2. 11. 2005 gelang es (nach 5 Monaten Rechenzeit auf einem Opteron-Cluster mit 80 2,2-GHz-Prozessoren), die 193-stellige "challenge number" RSA-640 in ihre beiden 97-stelligen Primfaktoren zu zerlegen.

Derzeit werden mindestens 768 bits für die Zahl n empfohlen (entspricht etwa 231 Dezimalstellen).

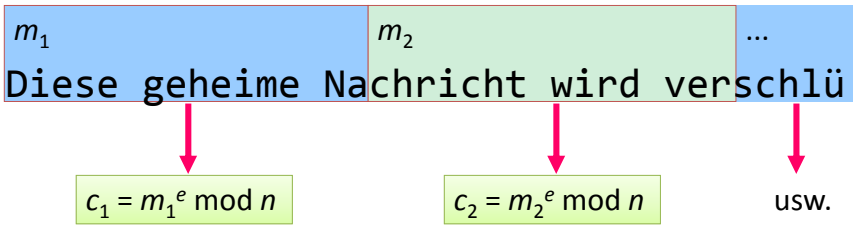
Algorithmus **Quadratisches Sieb**: Liefert einen Faktor von n:
- Suche $a_1, \ldots, a_k$ so dass alle $a_i^2 - n$ kleine Primfaktoren haben.
- Suche eine Teilmenge der $a_i^2 - n$, deren Produkt ein Quadrat x^2 ist.
- x := x mod n;
- y := (Produkt der a_i, die das Quadrat gebildet haben) mod n
- g := ggT(x-y, n)
- Falls g kein Teiler von n ist, suche weitere x und y (ggf. weitere a_i)

Das RSA-Verfahren

Praktische Anwendung

Aufteilung der Nachricht in Blöcke,
die noch in den Bereich 0 ... n-1 codiert werden können.

Beispiel: $n > 2^{128} = 2^{16 \cdot 8}$ (Blöcke zu 16 Bytes)



Schnellere Alternative

Die Nachricht wird konventionell chiffriert (z.B. AES),
nur der konventionelle Schlüssel wird mit RSA chiffriert.

RSA zum Nulltarif

Schlüsselerzeugung		mathGUIde
1	Wähle zwei Primzahlen p, q	p = nextPrime(2^500+rand(2^500)) q = nextPrime(2^500+rand(2^500))
2	n := pq, φ(n) = (p-1)(q-1)	n = p * q phi = (p-1) * (q-1)
3	d ∈ {1,..., φ(n)} mit ggt (d, φ(n)) = 1	d = phi while gcd(d,phi) != 1: d=rand(phi)
4	Mit erweitertem euklidischen Algorithmus x und y bestimmen: xd + y φ(n) = 1 e := x mod φ(n) ⇒ ed ≡ 1 (mod φ(n)) Öffentlicher Schlüssel: (e,n) Privater Schlüssel: (d,n)	t,x,y = gcdExt(d,phi) Erweiterter euklidischer Algorithmus e = x % phi (e*d) % phi nur zur Kontrolle

Kommunikation

Chiffrierung:	$m \rightarrow c = m^e \bmod n$	m = 666 c = expMod(m, e, n)
Dechiffrierung:	$c \rightarrow m = c^d \bmod n$	m = expMod(c, d, n)

RSA-Implementierung in mathGUIde

```
def createRsaKeys(bits=768):
    bits = bits//2 - 1
    # wähle zwei Primzahlen p, q
    p = nextPrime(2^bits+rand(2^bits))
    q = nextPrime(2^bits+rand(2^bits))
    n = p * q
    phi = (p-1)*(q-1)          # phi(n) (Eulersche Phi-Funktion)
    d = phi                    # wähle d aus 1..phi(n)
    while gcd(d,phi) != 1:     # mit d, phi(n) teilerfremd
        d = rand(phi)
    t,x,y = gcdExt(d,phi)
    e = x % phi
    publicKey = (e,n)
    privateKey = (d,n)
    return (publicKey, privateKey)
```

liefert ein RSA-Schlüsselpaar
Das Paar wird als Tupel ((e,n),(d,n)) zurückgegeben.

```
def rsaEncrypt(m, key):
    return pow(m, key[0], key[1])
```

Verschlüsselt m mit dem Schlüssel key = (e,n).

Beispiel

```
public, private = createRsaKeys()
m = 12345                      # Nachricht
c  = rsaEncrypt(m, public)     # verschlüsselte Nachricht
m1 = rsaEncrypt(c, private)
assert m1 == m
```