Algebraische Srukturen

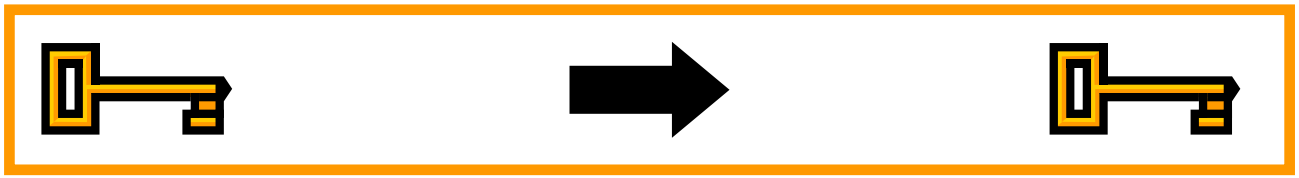
Kryptographie

M.B. Wischnewsky

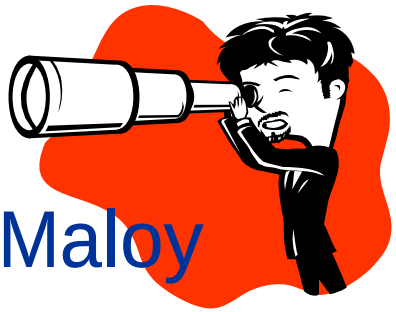
23.01.2007

Symmetrische Kryptographie

Alice



Bob

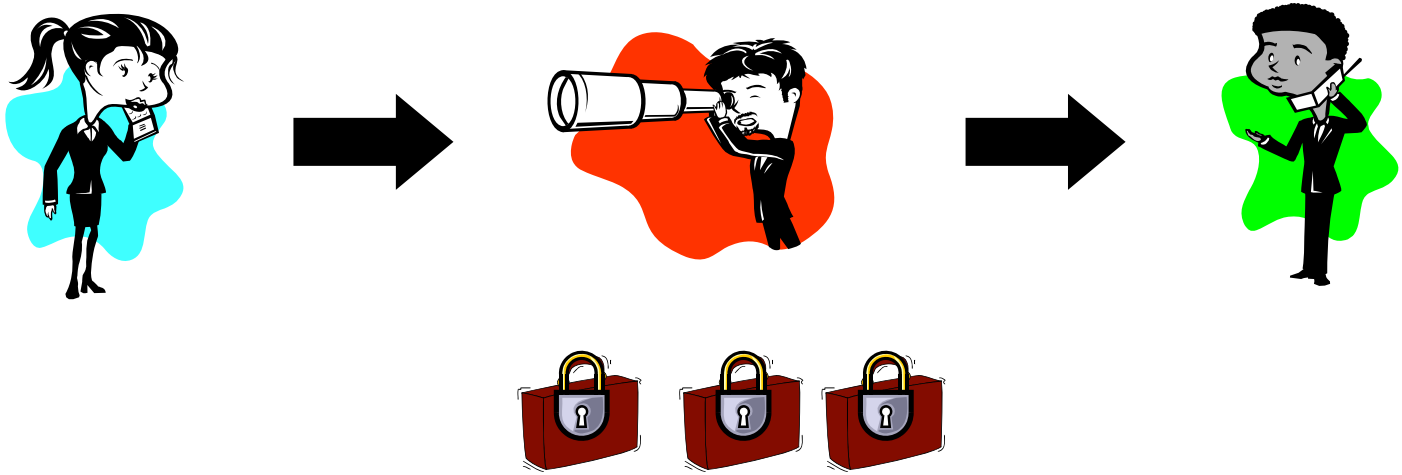


Maloy

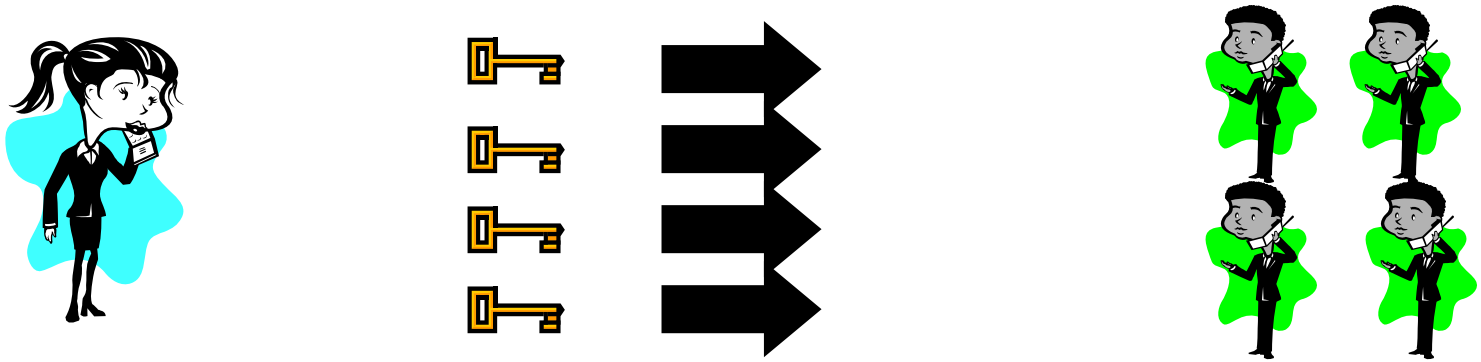
- Gemeinsames „Geheimnis“
- Schlüsselaustausch über sicheren Kanal
- Vorhängeschloß-Analogie

Symmetrische Kryptographie: Schwächen

- Replay Angriff

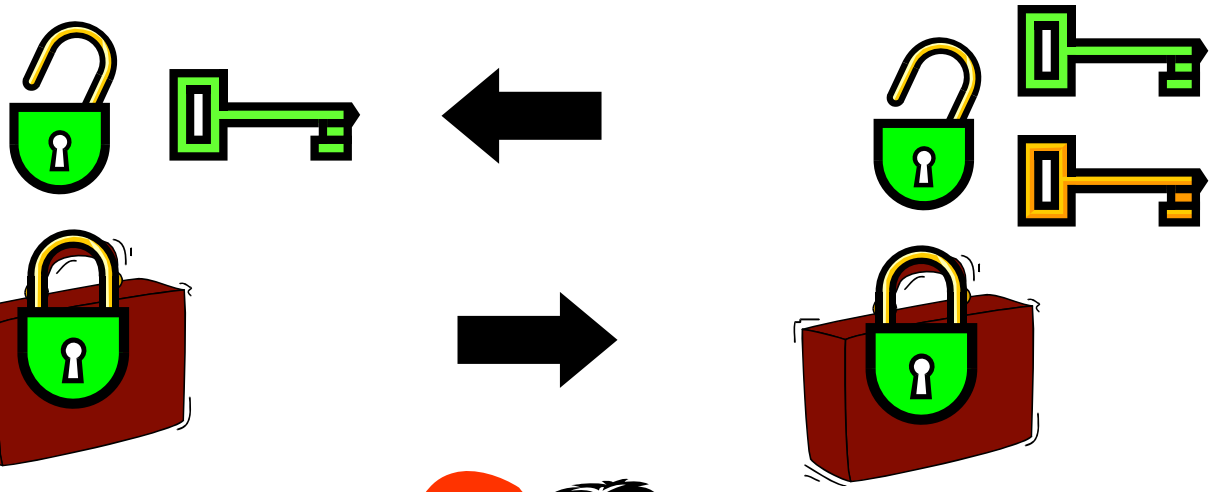


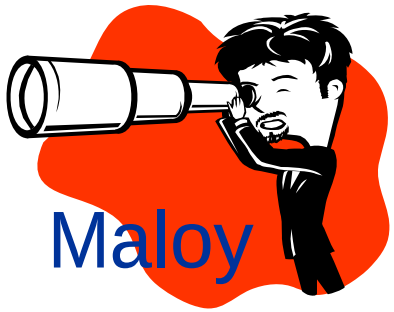
- Schlüsselaustausch?



Asymmetrische Kryptographie

Alice





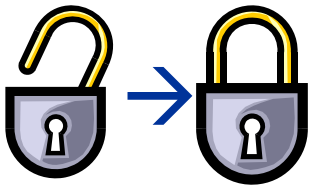
Maloy

Bob

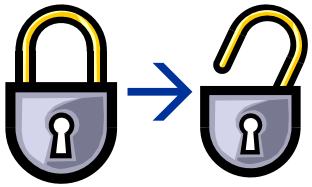
- Kein gemeinsames „Geheimnis“
- Privater Schlüssel zum Aufschließen
- Öffentlicher Schlüssel zum Zuschließen

Mathematische Vorhängeschlösser

- „asymmetrische“ Funktionen mit Hintertür:



$f(x) = y$ (einfach)



$f^{-1}(y) = x$ (schwierig)

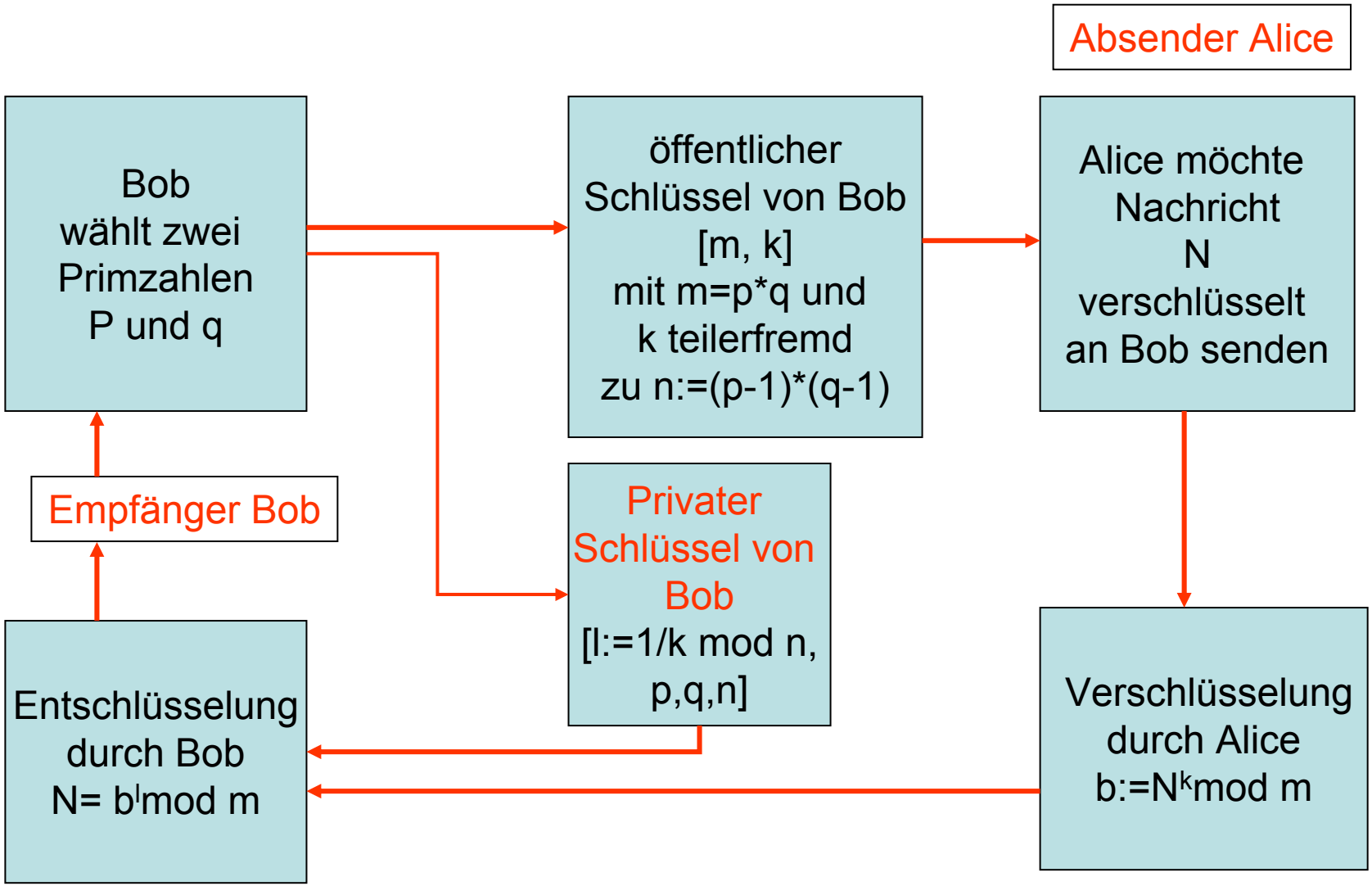
- Beispiele:
  - Faktorisierung (RSA)

    $31181593 * 11381633 = 354897447881369$
  - Potenzierung

    $42^3 = 42*42*42 = 74088$

    $3 = \ln_{42}(74088)$

RSA-Verschlüsselungsverfahren



RSA-Verschlüsselung

von Ronald Rivest, Adi Shamir und Leonard Adleman*

Ausgangslage

Der Absender Alice möchte eine geheime Nachricht an den Empfänger Bob übermitteln. Dabei besteht die Asymmetrie darin, dass Alice und Bob sich weder zu kennen noch irgend-wie geheim zu einem Schlüsselaustausch zu treffen brauchen.

*Rivest, R., Shamir, A., und Adleman, L.: *A method for obtaining Digital Signatures and Public Key Cryptosystems*, Comm. ACM, Vol. 21, Nr. 2, 1978

RSA-Verschlüsselungsverfahren

Vorbereitungen

- Bob wählt zwei „große“, d.h. etwa 200-stellige Primzahlen p, q, welche geheim gehalten werden.
- Bob berechnet m = p *q.
- Bob bestimmt eine natürliche Zahl k, die zu
  n =(p-1)*(q -1) teilerfremd ist.
- Bob übermittelt öffentlich die Daten m und k ; insbesondere Alice empfängt diese.

RSA-Verschlüsselungsverfahren

Vorgehen von Alice

- Alice stellt die zu übermittelnde Nachricht als natürliche Zahl N(0 < N< m) dar.
- Alice übermittelt öffentlich den Rest b bei der Division N^k durch m.

$$b = N^k \bmod m^*$$

*mod ist hier die Modulo-Funktion:

b mod m = r (r ist der Rest bei Division b/m)

Es gilt nach Vorlesung: b mod m = r → b mod m ≡ r , da zwei ganze Zahlen a und b genau dann kongruent modulo m sind, wenn sie bei Division durch m den gleichen Rest haben.

RSA-Verschlüsselungsverfahren

- Bob empfängt die Nachricht $b=N^k$ modm.
- Bob bestimmt ganze Zahlen l und y mit der Eigenschaft k *l = 1+ n *y.
- Bob berechnet c := b^lmod m (Rest bei der bei der Division b^l durch m.
- Bob hat damit die Nachricht entschlüsselt, denn es gilt c = a (kleiner Satz von Fermat).

Beispiel (maple): Bob

- p := 229;
- q := 389;

- m := p*q;
- k := 43; k ist eine beliebige ganze Zahl, die teilerfremd zu m ist.

(m,k) sind der öffentliche Schlüssel

- n := (p-1)*(q-1);
- l := 1/k mod n

l ist der private Schlüssel

Alice

- a := 666; Sei a der zu versendende Text.

- $b := a^k \bmod m$ (b wird an Bob geschickt)

Bob

> $c := b^l \bmod m$; c = 666 = a

RSA-Verschlüsselungsverfahren

- B wählt: $p = 229$, $q = 389$, also $m = p \cdot q = 89081$,
- $n = (p-1)\cdot(q-1) = 88464$; $k = 43$.
- B gibt bekannt: $m = 89081$, $k = 43$.
- Geheime Nachricht: $N = 666$.
- A übermittelt: $b = N^k \bmod m$, d.h. $b = 42709$.
- B bestimmt ganze Zahlen l, y mit
  $k \cdot l = 1 + n \cdot y$, d.h $[k][l] = [1] \bmod n$
- nämlich $l = 67891$, $y = 33$.
- B entschlüsselt: $c = b^l \bmod m$, d.h.
  $c = 666 = a$.

RSA-Verschlüsselungsverfahren

 # Das RSA Public Key Kryptographische Schema (Maple)
#
Ausgangspunkt: Zwanzigstellige Zufallszahlen.
Bob generiert einen öffentlichen und einen privaten Schlüssel.
#

```
> restart;
> N:=rand(10^19..10^20 -1)();
> M:=rand(10^19..10^20 -1)();
```

N := 19669081321110693270
M := 23073697474256143563

```
> p:=nextprime(N);
> q:=nextprime(M);
```

p := 19669081321110693313
q := 23073697474256143681

```
> m:=p*q;
```

m := 453838431999850498062034427355653905153

RSA-Verschlüsselungsverfahren

length(m);

39

> n:=(p-1)*(q-1);

n := 45383843199985049801929164856028706816O

> k:=rand(10^19..10^20-1)();

k := 819526553100754871 63

> l:=1/k mod n;

l := 188458591620039186939711640707989662067

Überprüfe ob l*k mod (n) =1

➢l*k mod n; l*k mod n = 1

Der öffentliche Schlüssel der zu Alice geschickt wird

> Öffentlicher_Schlüssel:=[m,k];

Öffentlicher_Schlüssel := [

45383843199985049806203442735565390515 3,

819526553100754871 63]

RSA-Verschlüsselungsverfahren

Der private Schlüssel von Bob lautet:
> Privater_Schlüssel:=[p,q,l,n];

Privater_Schlüssel := **[19669081321110693313, 23073697474256143681,**
188458591620039186939711640707989662067,
45383843199985049801929164856028706816O]

Alice will einen Text an Bob verschlüsselt senden
> **klar_Text:="Das folgende Text ist streng geheim:**
> > Die Mathematikveranstaltung für Informatiker gibt eine spannende
> Einführung in die Grundbegriffe der Algebra";
klar_Text := "Das folgende Text ist streng geheim:\n\nDie Mathem\
atikveranstaltung für Informatiker gibt eine spannende Ein\
führung in die Grundbegriffe der Algebra"

> List1:=convert(klar_Text,bytes);

List1 := [68, 97, 115, 32, 102, 111, 108, 103, 101, 110, 100, 101, 32, 84, 101, 120, 116, 32, 105, 115, 116, 32, 115, 116, 114, 101, 110, 103, 32, 103, 101, 104, 101, 105, 109, 58, 10, 10, 68, 105, 101, 32, 77, 97, 116, 104, 101, 109, 97, 116, 105, 107, 118, 101, 114, 97, 110, 115, 116, 97, 108, 116, 117, 110, 103, 32, 102, 252, 114, 32, 73, 110, 102, 111, 114, 109, 97, 116, 105, 107, 101, 114, 32, 103, 105, 98, 116, 32, 101, 105,110, 101, 32, 115, 112, 97, 110, 110, 101, 110, 100, 101, 32, 69, 105, 110, 102, 252, 104, 114, 117, 110, 103, 32, 105, 110, 32, 100, 105, 101, 32, 71, 114, 117, 110, 100, 98, 101, 103,114, 105, 102, 102, 101, 32, 100, 101, 114, 32, 65, 108, 103, 101, 98, 114, 97]

RSA-Verschlüsselungsverfahren

List2:=convert(List1,base, 256,m);
List2 := [259336524063177009973934606631916658340,
395553762530464078955923815356029755507,
279088033751339392783027572364796741608,
415415239491770292342329813300087132 15,
447695460678539772311420569266273497310,
383269262836036632656658032730014095486,
427083017851967114632436563418376362085,
643864866235671508434911367703416855 47,
145073510609882404503647932605061646606, 1868]

Nun wenden wir die Abbildung $x \rightarrow x^k \bmod (m)$ auf jede Zahl in List2 an Dies wird mit dem Befehl map durchgeführt.
Beispiel:
> L1:=[1,2,3,4,5];
> L2:=map(f,L1); L2 := [f(1), f(2), f(3), f(4), f(5)]
definiere f als Funktion
> f(x) = x^2; :
- f:=x->x^2;
- > L3:=map(f,L1); L3 := [1, 4, 9, 16, 25]

RSA-Verschlüsselungsverfahren

Nun definieren wir f neu durch
> f(x,k,m) = x^k mod (,m);# . Wir benützen hierzu die Maplefunktion Power(x,k) mod m.
#
> f:=(x,k,m)->Power(x,k) mod m;

f := (x, k, m) -> Power(x, k) mod m

Wir illustrienen die Funktionen an den Wertepaaren k,m aund l,m Die
zugehörigen Funktionen sind invers zueinander..
Beispiel:
> x:=5;
> a:=f(x,k,m);
> b:=f(a,l,m);

x := 5

a := 430123273283798962924241316297043948755

b := 5

Nun wenden wir dies auf List2 an, um den verschlüsselten Text von
Alice an Bob zu erhalten.

Nun wenden wir dies auf List2 an, um den verschlüsselten Text von
Alice an Bob zu erhalten.

> **verschlüsselter_Text:=map(f, List2,k,m);**

```
  verschlüsselter_Text := [14932613597501909383524835429918837850 2,
        41981644517435322124251853421131969942 9,
        30293399410064754640782860324136844510 8,
        18361101284774199852444223294686244808 4,
        12616501699096898615046908960402338391 0,
        20908067682408087441213580804423063763 9,
        14748598282882001663394649375496340013 9,
        26894383893813313213135993375022826666 4,
        95082440524816493065079192195982350777,
        26782623652457102070524122055821564148 8]
```

> **entschlüsselt1:=map(f,verschlüsselter_Text,l,m);**

```
  entschlüsselt1 := [25933652406317700997393460663191665834 0,
        39555376253046407895592381535602975550 7,
        27908803375133939278302757236479674160 8,
        41541523949177029234223298133000871321 5,
        44769546067853977231142056926627349731 0,
        38326926283603663265665803273001409548 6,
        42708301785196711463243656341837636208 5,
        64386486623567150843491136770341685547,
        14507351060988240450364793260506164660 6, 1868]
```

RSA-Verschlüsselungsverfahren

entschlüsselt2:=convert(entschlüsselt1,base,m,256);

```
  entschlüsselt2 := [68, 97, 115, 32, 102, 111, 108, 103, 101, 110,
        100, 101, 32, 84, 101, 120, 116, 32, 105, 115, 116, 32, 115,
        116, 114, 101, 110, 103, 32, 103, 101, 104, 101, 105, 109, 58,
        10, 10, 68, 105, 101, 32, 77, 97, 116, 104, 101, 109, 97, 116,
        105, 107, 118, 101, 114, 97, 110, 115, 116, 97, 108, 116, 117,
        110, 103, 32, 102, 252, 114, 32, 73, 110, 102, 111, 114, 109,
        97, 116, 105, 107, 101, 114, 32, 103, 105, 98, 116, 32, 101,
        105, 110, 101, 32, 115, 112, 97, 110, 110, 101, 110, 100, 101,
        32, 69, 105, 110, 102, 252, 104, 114, 117, 110, 103, 32, 105,
        110, 32, 100, 105, 101, 32, 71, 114, 117, 110, 100, 98, 101,
        103, 114, 105, 102, 102, 101, 32, 100, 101, 114, 32, 65, 108,
        103, 101, 98, 114, 97]

# Schließlich erhält Bob den Klartext von Alice zurück.
> entschlüsselt3:=convert(entschlüsselt2,bytes);

  entschlüsselt3 := "Das folgende Text ist streng geheim:\n\nDie M\
        athematikveranstaltung für Informatiker gibt eine spannend\
        e Einführung in die Grundbegriffe der Algebra"
```

Wiederholungen

1) Der erweiterte euklidische Algorithmus

Bei der RSA-Verschlüsselung benötigt man die zu k inverse Zahl l so dass gilt:
$k \cdot l \equiv 1 \bmod n$ (wobei $n = (p-1)\cdot(q-1)$ ist).
Anders ausgedrückt. Es werden zwei ganze Zahlen l und x ermittelt für die gilt: $l \cdot k + x \cdot m = 1$.
Die Koeffizienten l und x können mit Hilfe des erweiterten euklidischen Algorithmus ermittelt werden.

Satz (über den größten gemeinsamen Teiler).
Sind $a, b \in Z$ nicht beide Null und ist t=ggT(a,b) ihr größter gemeinsamer Teiler so gilt:
1. Es gibt ganze Zahlen α und β mit $\alpha \cdot a + \beta \cdot b = t$.
2. Teilt $d \in Z$ sowohl a als auch b, so teilt d notwendig den größten gemeinsamen Teiler von a und b.

Wiederholungen

1) Der erweiterte euklidische Algorithmus

Der **erweiterte euklidische Algorithmus** liefert uns den größten gemeinsamen Teiler t=ggT(a,b) zweier ganzer Zahlen a und b **und** passende ganze Zahlen α und β mit $\alpha \cdot a + \beta \cdot b = t$.

Der euklidische Algorithmus ist durch folgende **Rekursionsformel** definiert
ggT(a,0) = a und ggT(a,b) = ggT(b,a mod b) für b ≠0

Beispiel:
ggT(100,35)=ggT(35,100 mod 35)= ggT(35,30)
ggT(35,30) =ggT(30,35 mod 30) = ggT(30,5)
ggT(30,5) =ggT(5,30 mod 5) = ggT(5,0) = 5

Wiederholungen

1) Der erweiterte euklidische Algorithmus

Seien zwei ganze Zahlen a,b teilerfremd, d.h. ggT(a,b) = 1, dann gibt es ganze Zahlen α und β mit $\alpha \cdot a + \beta \cdot b = 1$.

2) Der Ring Z_n der ganzen Zahlen modulo n

Wir haben eine Addition und eine Multiplikation in Z_n
[a] + [b] = [a+b], und
a]*[b] = [a*b], wobei
[x] := {z $\in$Z; z ~ x }
Es gilt: z ~ x $\leftrightarrow$ z – x $\in$Zn, d. h. n|(z-x).
Schreibweise: z ~ x $\leftrightarrow$ z$\equiv$ x mod(n)
Es gilt also z ~ x $\leftrightarrow$ es existiert eine ganze Zahl k mit

$$z = x + k*n$$

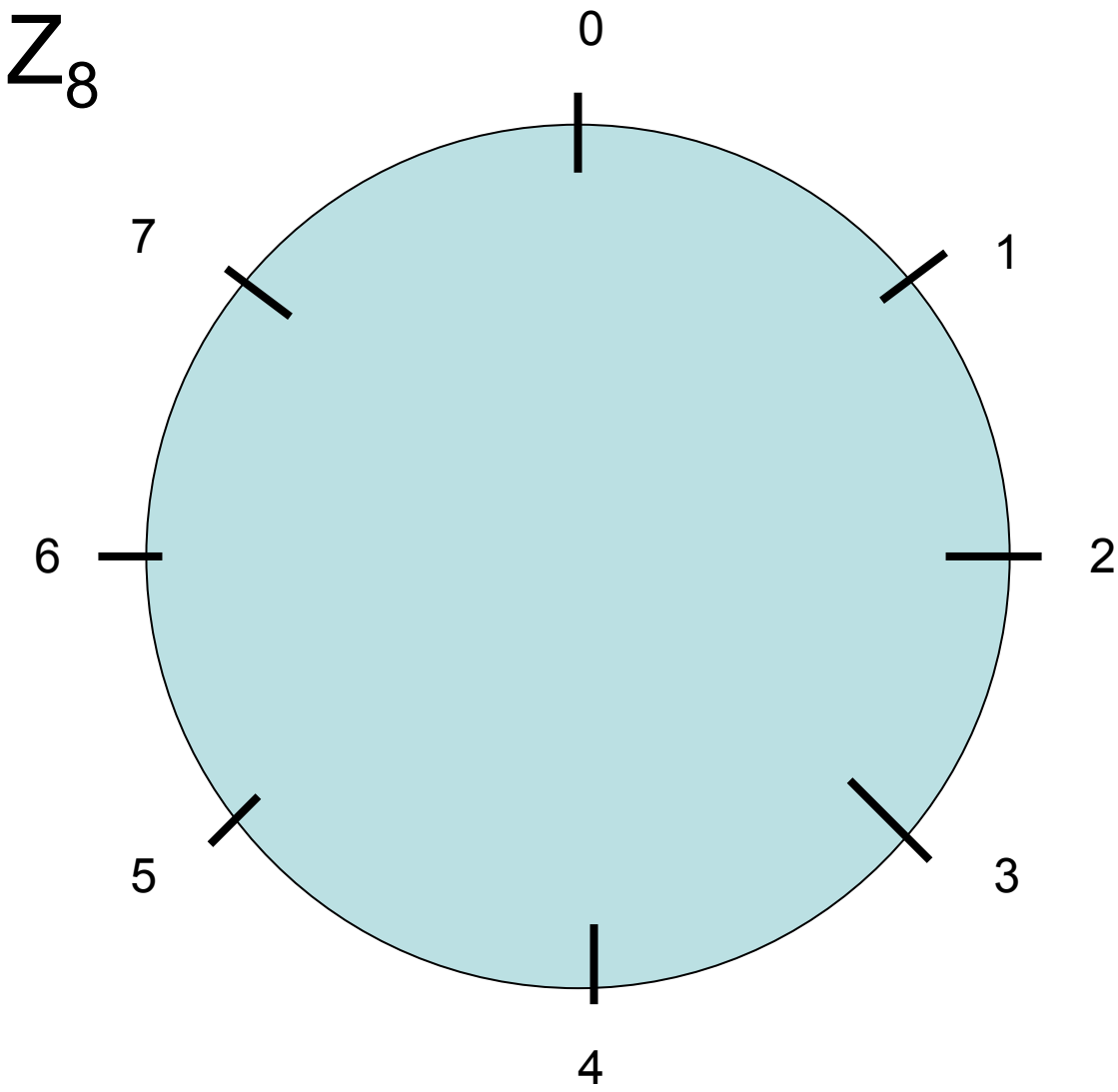
Die Modulo-Funktion:

Definition:
-mod- : ZxZ → Z (a,b) → a mod b = r
(r ist der Rest bei Division a/b)

Es gilt nach Vorlesung: a mod b = r → r ≡ a mod b , da gilt
Satz: Zwei ganze Zahlen x und y sind genau dann kongruent modulo b, wenn sie bei Division durch b den gleichen Rest haben.

Gilt a mod b = r, so ist der Rest r bei Divion a/b natürlich kongruent zu a, denn es gilt: r mod b = r und a mod b = r

Beispiel: Rechnen modulo 8



[4] + [5]= [9] = [1],

Der Ring Z_n der ganzen Zahlen modulo n

- **Satz** Z_n ist mit der Addition und Multiplikation ein kommutativer Ring mit Einselement.

Bemerkung: Sei R ein Ring mit Einselement 1.

Definition. $a \in R$ heißt **invertierbar**, wenn es ein $b \in R$ gibt mit $a*b = 1$

Hieraus folgt:$[a] \in Z_n$ ist invertierbar, genau dann wenn es eine ganze Zahl b gibt mit $[a] * [b] = [1]$, d.h. $a*b$ **~ 1**

bzw. a*b $\equiv 1 \bmod(n)$. Hieraus folgt: Es gibt eine ganze Zahl k mit $a*b = 1 + k*n$, d. h. $ggT(a,n) = 1$.

Satz :$[a] \in Z_n$ ist invertierbar, genau dann wenn $ggT(a,n)=1$.

RSA Algorithmus

p, q große Primzahlen

$m := p \cdot q$

$n := (p-1) \cdot (q-1)$

Wähle $k \in \mathbb{Z}$ mit k teilerfremd zu n

Dann gilt $ggT(k, n) = 1 \Leftrightarrow$

$\exists\ l \in \mathbb{Z}$ $\quad k \cdot l \equiv 1 \bmod (n)$

öffentl. Schlüssel $[m, k]$

privater " $[m, l, p, q]$

Algorithmus Sei N eine Nachricht $1 \leq N < m$

Alice	$S := N^k \bmod m$ (verschlüsselt)
Bob	$N' := S^l \bmod m$

Dann gilt $N' = S^l \bmod m = N$

Beweis Der Beweis erfolgt in 4 Schritten.

① $N' = S^l \bmod m = (N^k \bmod m)^l \bmod m$

$\overset{*}{=} N^{kl} \bmod m$

Zu zeigen $N = N^{kl} \bmod m$

*) $a^k \bmod m = (a \bmod m)^k$

RSA Algorithmus

(2) k teilerfremd zu n $\Rightarrow$ Es gibt $l, x \in \mathbb{Z}$

mit $k \cdot l = 1 + x(p-1)$ $(n=(p-1)(q-1))$

Kleiner Fermat: p Primzahl, $a \in \mathbb{Z}$

$\Rightarrow a^{p-1} \equiv 1 \bmod p$

$N^{p-1} \equiv 1 \bmod p \Leftrightarrow \boxed{1 = N^{p-1} \bmod p}$

$N^{kl} \bmod p = N^{1+x(p-1)} \bmod p =$

$N \left(N^{p-1}\right)^x \bmod p = N \bmod p \left(N^{p-1} \bmod p\right)$

$= N \bmod p$ $\quad \approx 1$

$\Rightarrow N \equiv N^{kl} \bmod p$

RSA Algorithmus

③ analog (Vertauschung p, q)

$$N \equiv N^{kl} \bmod q$$

④ $N \equiv N^{kl} \bmod p \Rightarrow p \mid N - N^{kl}$

$N \equiv N^{kl} \bmod q \Rightarrow q \mid N - N^{kl}$

$\Rightarrow pq \mid N - N^{kl} \Rightarrow$

$N \equiv N^{kl} \bmod pq = N^{kl} \bmod m$

Da $N < m \Rightarrow N = N^{kl} \bmod m$ (Rest)

$\Rightarrow N' = N$. □