RSA-Verfahren, Public-Key-Verschlüsselung

Kryptographie mit MuPAD,

Prof. Dr. Dörte Haftendorn, Mathematik mit MuPAD 4.02,

(ex. in 2.5 vom Sept 99 Nov.02 und in 3.11 Sept. 05) Feb.07

http://haftendorn.uni-lueneburg.de **www.mathematik-verstehen.de**

##

----------eigene Zahlentheorie Ergänzungen----------

Im Dateimenu bei "Eigenschaften" sehen die beien Prozeduren zur Umwandlung von Text in Zahl und Zahl in Text, daher können sie hier ausgeführt werden.

----------eigene Zahlentheorie Ergänzungen----------

Anton bereitet seine Schlüssel vor.

```
p:=numlib::prevprime(floor(sqrt(1822*10^50))://Exponent z.B.auch 20
```

```
q:=numlib::prevprime(floor(sqrt(27*10^50)));
```

519615242270663188058233 13

```
//p:=11;q:=13;
```

```
n:=p*q;
```

221797204671294268320361329839012938986021176757039 01

```
ph:=(p-1)*(q-1);
```

221797204671294268320361325050908502632749092208629 12

```
repeat
r:= random(2..ph): e:=r():
  until gcd(ph,e)=1 end_repeat:e;
```

205068709127346515314857867405809035561697975043319 27

```
d:=op(igcdex(ph,e),3): // letztes Element euklid. Algorithmus
```

```
if (d<0) then d:=d+ph: end_if:d; //Korrektur bei negativem d
```

105970508438549193695486431008420269606378489387909 51

Das ist Antons geheimer Schlüssel. Der Öffentlichkeit gibt er e und n bekannt.

```
e;n;
```

205068709127346515314857867405809035561697975043319 27

221797204671294268320361329839012938986021176757039 01

Berta will Anton einen Text senden, den nur Anton lesen kann.

----------eigene Zahlentheorie Ergänzungen----------

Im Dateimenu bei "Eigenschaften" sehen die beien Prozeduren zur Umwandlung von Text in Zahl und Zahl in Text, daher können sie hier ausgeführt werden.

----------eigene Zahlentheorie Ergänzungen----------

```
m:=txToZoo("Montag im Medley");  //ASCII-30, zweistellig
```

```
[77, 111, 110, 116, 97, 103, 32, 105, 109, 32, 77, 101, 100, 108, 101, 121
```

478180866773027579024771707871 91

```
gcd(m,n); // soll 1 sein
```

1

```
c:=powermod(m,e,n);
```

237848029414497557102732162788324136308556081917516 7

Anton empfängt diesen verschlüsselten Text und wandelt ihn in Klartext um.

```
m:=powermod(c,d,n);
```

478180866773027579024771707871 91

```
47818086677302757902477170787191
```

```
zooToTx(m) //Zweierpakete +30->ASCII
[47, 81, 80, 86, 67, 73, 2, 75, 79, 2, 47, 71, 70, 78, 71, 91]
[77, 111, 110, 116, 97, 103, 32, 105, 109, 32, 77, 101, 100, 108, 101, 121
```

"Montag im Medley"

Anton signiert einen der Öffentlichkeit präsentierten Text

```
M:=txToZoo("Dies sagt euch Anton.");
[68, 105, 101, 115, 32, 115, 97, 103, 116, 32, 101, 117, 99, 104, 32, 65,

   116, 111, 110, 46]
```

387571850285677386027187697402358086818016

```
sig:=powermod(M,d,n);
```

150534459759463101059954929009825908896734235141704 80

Anton stellt den Text und sig öffentlich aus. Eigentlich wendet er auf M vorher noch eine Hashfunktion an,
die öffentlich bekannt ist und M auf 128 Bit reduziert. Hier ist h(M)=M.

Berta will prüfen, ob das wirklich Antons unveränderter Text ist.

```
MB:=M: //Berta berechnet ebenfalls aus dem erhaltenen MB das h(M),
      //hier ist h(MB)=MB. Unsd sie bildet ein Mtest aus der Signatur.
```

```
Mtest:=powermod(sig,e,n);
```

387571850285677386027187697402358086818016

```
if (modp(MB,n)=modp(Mtest,n)) then print("Anton hat wirklich unterschrieben")
else print("Vorsicht, das ist nicht Antons Original!") end_if;
"Anton hat wirklich unterschrieben"
```

Wenn aber nun MisterX verändert hat

```
MB:=M+1
```

387571850285677386027187697402358086818017

```
if (modp(MB,n)=modp(Mtest,n)) then print("Anton hat wirklich unterschriebe
else print("Vorsicht, das ist nicht Antons Original!") end_if;
"Vorsicht, das ist nicht Antons Original!"
```