RSA-Verfahren, Public-Key-Verschlüsselung

Kryptographie mit MuPAD, Prof. Dr.Dörte Haftendorn, Sept.99, Nov 02, Sept. 05

- `delete PACKAGEPATH:endml:=strmatch(NOTEBOOKPATH,"mathe-lehramt", Index)[2]: gesamtpackpfad:=substring(NOTEBOOKPATH,1..endml+1).pathname("computer","mupad", "packages"): PACKAGEPATH:=gesamtpackpfad,PACKAGEPATH://Tipps zu Packages siehe unten auf der Seite.`
- **`package("zahltheo", Forced):zahltheo::init():export(zahltheo):`**

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

-

Anton bereitet seine Schlüssel vor.

- **`p:=numlib::prevprime(floor(sqrt(1822*10^50)); //Exponent z.B.auch 200`**

  426848919408260889649017677

- **`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;
  ```

  543181035955889586801251014790675614875822131493300 7

- **`d:=op(igcdex(ph,e),3): // letztes Element euklid. Algorithmus`**
- **`if (d<0) then d:=d+ph: end_if:d; //Korrektur bei negativem d`**

  132307348725174434100469933552174206877238047166887 51

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

- `e;n;`

  543181035955889586801251014790675614875822131493300 7

  221797204671294268320361329839012938986021176757039 01

  ```

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

- `m:=txToZoo("Montag im Medley");`

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

47818086677302757902477170787191

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

1

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

211812519735596349852033514013658497674160593927888873

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

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

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,

   110, 116, 111, 110, 46]
```

387571850285677386027187697402358086818016

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

181778485770045618809513011646355594700932540536235 36

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);**

  3875718502856773860271876974023580868180 16

- **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**

  3875718502856773860271876974023580868180 17

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

  "Vorsicht, das ist nicht Antons Original!"

-

Tipps zu Packages: Einzelne *.mu-Dateien kann man auch mit dem Menu Notebook -->Einlesen -->*.mu hinzufügen. Auf dieser Site liest der oben genannte Befehl alle eigene Funktionen der Zahlentheorie.

-