RSA für mittelgroße Primzahlen

RSA−Verfahren, Version mit großen Primzahlen   Haftendorn Okt 2011

Anton möchte, dass jeder ihm Nachrichten schicken kann, die nur er selbst lesen kann.

Niemand, der die Kommunikation abfängt, soll eine Chance haben, den Klartext herauszubekommen.

Anton bereitet seine Schlüssel vor: Er wählt zwei Primzahlen

p:=**kry\nextprime**(randInt(100000,1000000)) ▸ 357431   und

q:=**kry\nextprime**(randInt(100000,1000000)) ▸ 542783 .

10^{14} *ist maximale Größe für randint. Da das Produkt verwendet wird, gehe ich hier bis etwa* 10^{7}

Das Produkt wird der erste Teil seines öffentlichen Schlüssels **n**:=**p**·**q** ▸ 194007470473 .

In der Gruppe Z*(n) wird später potenziert, daher braucht er die Ordnung von Z*(n).

phi:=(**p**−1)·(**q**−1) ▸ 194006570260 . Nun wählt er ein "gutes" **e** aus Z*(phi).

e:=randInt(50000,**phi**−2) ▸ 44447167471   und prüft sofort
li:=**kry\ggte**(**e**,**phi**) ▸ [1 −55726811009 12767087724]

Wenn hier vorn keine 1 steht, schickt er den Befehl für das Zufalls−e nochmal ab.

"Schlechte" **e** sind solche mit zu kleiner Ordnung. Das passiert bei großen Beispielen nicht so leicht.

Es ließe sich auch schwer prüfen. Ordnungen 2 bis 5 ausprobiert, keine 1 dabei ist gut:
seq(**kry\pmod**(**e**,*k*,**n**),*k*,2,5) ▸ {180697569452,36671586154,72001127724,51959103257}

Zu diesem **e** braucht er das Inverse in Z*(phi). Es ist das zweite Element der Liste li (wenn vorn 1 steht)
d:=mod(**li**[1,2],**phi**) ▸ 138279759251 , modulo phi genommen. Falls es negativ war, ist ein n aufaddiert worden.

1.1

Probe: mod(**e·d,phi**) ▸ 1 . Hier muss 1 stehen. Sein d hält er geheim.

Auf **Antons Internetseite steht nun für jeden zu lesen**:

Mein öffentlicher RSA−Schlüssel ist das Zahlenpaar [**e** **n**] ▸ [44447167471 194007470473]

Anwendungsphase

Berta will Anton eine Nachricht senden **klar**:="Hanse" ▸ Hanse

m:=**wort2zahl**(**klar**) ▸ 4469828773 Ist **m**<**n** ▸ true True ist gut.

c:=**kry\pmod**(**m**,**e**,**n**) ▸ 126893329615 Dies erhält Anton.

Entschlüsselung, Anton kann die Nachricht lesen

mm:=**kry\pmod**(**c**,**d**,**n**) ▸ 4469828773 **liest**:=**zahl2wort**(**mm**) ▸ Hanse

Zur Erinnerung es war **m** ▸ 4469828773

Berta darf ihre Nachricht nicht größer als n machen. **klarf**:="Hansen" ▸ Hansen

mf:=**wort2zahl**(**klarf**) wäre so eine falsche Nachricht. **mf**<**n** ▸ false

cf:=**kry\pmod**(**mf**,**e**,**n**) ▸ 47973612975 Dies erhält Anton.

Entschlüsselung, Anton kann die Nachricht nicht !!!!!! lesen

mmf:=**kry\pmod**(**cf**,**d**,**n**) ▸ 58967936436 Zur Erinnerung es war **mf** ▸ 446982877382 Es ist mod(**mf**,**n**) ▸ 58967936436 . Aha wir sehen den Zusammenhang,

Anton sieht wirres Zeug: **zahl2wort**(**mmf**) ▸ !u_y\@

1.2

Digitale Signatur mit RSA

Anton will in seinem Internet eine Botschaft signieren, also digital unterschreiben.

Wer diese Klartext−Botschaft liest soll prüfen können, ob sie wirklich von Anton ist.

Mister X könnte ja evt. als Spion im Rechenzentrum Antons Botschaft manipuliert haben.

−−−−−−−−−−−−−−−−−−

Signatur Anwendungsphase

Seine Klartext−Botschaft sei: **klars**:="Egon? " **ms**:=**wort2zahl**(**klars**)

Anton berechnet **sig**:=**kry\pmod**(**ms**,**d**,**n**) ▸ 77897632562 und stellt die "Signatur" daneben.

−−−−−−−−−−−−−−−−−−

Signatur Prüfungsphase

Berta liest die beides und prüft: **mp**:=**kry\pmod**(**sig**,**e**,**n**) ▸ 4175838235

Wenn sie auf diese Weise auch wieder die Klartext−Nachricht **zahl2wort**(**ms**) ▸ Egon? erhält, hat niemand Antons Site manipuliert.

−−−−−−−−−−−−−−−−−−

Info für Informatik−Studierende: Es werden, um das Datenvolumen kleiner zu halten, noch sogenannte Hash−Funktionen zwischengeschaltet (siehe Wikipedia und www.mathematik−verstehen.de). Das ist für das Verstehen aber unerheblich.

1.3

* wort2zahl 1/8

```
Define LibPub wort2zahl(klar)=
Func
©(klar)–>Zahl mit ASCII–28
Local z,w,i
w:=klar: z:=0
For i,1,dim(klar)
z:=z·100+ord(left(w,1))–28
w:=mid(w,2)
EndFor
Return z
EndFunc
```

"zahl2wort" erfolgreich gespeichert

```
Define LibPub zahl2wort(zahl)=
Func
© (zahl)–>Wort
Local i,z,w
w:=""
 z:=zahl
While z>0
w:=char(mod(z,100)+28)&w
z:=floor(z/100)
EndWhile
```

wort2zahl("hallo ") ▸ 7669808083

wort2zahl("Dora") ▸ 40838669

ord("h") ▸ 104   ord("h")–28 ▸ 76

zahl2wort(7669808083) ▸ hallo

wort:="erz " ▸ erz   dim(wort) ▸ 3   mid(wort,2) ▸ rz   left(wort,1) ▸ e

char(104)&wort ▸ herz

1.4