Diffie-Hellman + OnetimePad

Kryptografische Verfahren: Diffie-Hellman-Schlüssel-Vereinbarung Haftendorn Okt 2011

erg:=**diffi**(127,56,10,77) ▸ $\begin{bmatrix} \text{"Sie verabreden p "} & 127 & \text{"Sie verabreden g "} & 56 \\ \text{"Anton sendet "} & 30 & \text{"Berta sendet "} & 90 \\ \text{"Antons Schüssel "} & 37 & \text{"Bertas Schlüssel "} & 37 \end{bmatrix}$ isPrime(**erg**[1,2]) ▸ true

Diese Datei zeigt das Diffie-Hellman-Verfahren.

Die **Umwandlung von Zahlen in Ziffernlisten und zurück**

zahl2li(192837465) ▸ {1,9,2,8,3,7,4,6,5} **li2zahl**({1,9,2,8,3,7,4,6,5}) ▸ 192837465

und das OneTimePad-Verfahrenfür Ziffernlisten mit verschlüsseln und entschlüsseln.

onetimepad({1,2,3,4,5,6,7},{1,9,2,8,3,7,4,6,5}) ▸ {2,1,5,2,8,3,1}

onetimeinv({2,1,5,2,8,3,1},{1,9,2,8,3,7,4,6,5}) ▸ {1,2,3,4,5,6,7}

Man kann auch alles kombinieren:

diffiehell:=**diffi**(**kry\nextprime**(randInt(10000000000,100000000000000),randInt(10000000,10000000000),randInt(10000000,10000000000),randInt(10000000,10000000000))

▸ $\begin{bmatrix} \text{"Sie verabreden p "} & 96166031337767 & \text{"Sie verabreden g "} & 8671702491 \\ \text{"Anton sendet "} & 40899640805182 & \text{"Berta sendet "} & 35377272962276 \\ \text{"Antons Schüssel "} & 43662590709547 & \text{"Bertas Schlüssel "} & 43662590709547 \end{bmatrix}$

s:=**diffiehell**[3,2] ▸ 43662590709547 **sli**:=**zahl2li**(**s**) ▸ {4,3,6,6,2,5,9,0,7,0,9,5,4,7}

otp:=**onetimepad**(**zahl2li**(112233445566),**sli**) ▸ {5,4,8,8,5,8,3,4,2,5,5,1}

oti:=**onetimeinv**(**otp**,**sli**) ▸ {1,1,2,2,3,3,4,4,5,5,6,6} Ende: **li2zahl**(**oti**) ▸ 112233445566

Die Message 112233445566kommt am Ende wirder heraus.

1.1

diffi(13,2,5,4) ▸

"Sie verabreden p "	13	"Sie verabreden g "	2
"Anton sendet "	6	"Berta sendet "	3
"Antons Schüssel "	9	"Bertas Schlüssel "	9

diffi(123457,5678,1023,5001) ▸

"Sie verabreden p "	123457	"Sie verabreden g "	5678
"Anton sendet "	42332	"Berta sendet "	80229
"Antons Schüssel "	68591	"Bertas Schlüssel "	68591

diffi(**kry\nextprime**(75247388436868349139),55665526616813868,10248276867348633,57657657001) ▸

"Sie verabreden p "	75247388436868349167	"Sie verabreden g "	55665526616813868
"Anton sendet "	48122236671797693468	"Berta sendet "	8210090613061314686
"Antons Schüssel "	69377184492155382014	"Bertas Schlüssel "	69377184492155382014

10^{14} ist maximale Größe für randint, 10^{14} funktioniert nicht als Eintrag (Fehler gemeldet Okt. 2011)

diffi(**kry\nextprime**(randInt(10000000000,100000000000000)),randInt(10000000,10000000000),
randInt(10000000,10000000000),randInt(10000000,10000000000)) ▸

"Sie verabreden p "	6758801582209	"Sie verabreden g "	3849525901
"Anton sendet "	6597997620719	"Berta sendet "	6331204062781
"Antons Schüssel "	6631609941997	"Bertas Schlüssel "	6631609941997

Spielwiese

erg:=**diffi**(127,56,10,77) ▸

"Sie verabreden p "	127	"Sie verabreden g "	56
"Anton sendet "	30	"Berta sendet "	90
"Antons Schüssel "	37	"Bertas Schlüssel "	37

isPrime(**erg**[1,2]) ▸ true

1.2

diffi 4/10

```
Define LibPub diffi (p,g,a,b)=
Func
Local α,β,mat
α:=kry\pmod(g,a,p): β:=kry\pmod(g,b,p)
mat:=newMat(3,4)
mat[1,1]:="Sie verabreden p ":mat[1,3]:="Sie verabreden g "
mat[1,2]:=p:mat[1,4]:=g:
mat[2,1]:="Anton sendet ":mat[2,3]:="Berta sendet "
mat[2,2]:=α:mat[2,4]:=β:
mat[3,1]:="Antons Schüssel ":mat[3,3]:="Bertas Schlüssel "
mat[3,2]:=kry\pmod(β,a,p):mat[3,4]:=kry\pmod(α,b,p)
Return mat
EndFunc
```

```
diffi(19,13,7,10)
```

"Sie verabreden p "	19	"Sie verabreden g "	13
"Anton sendet "	10	"Berta sendet "	6
"Antons Schüssel "	9	"Bertas Schlüssel "	9

```
diffi(kry\nextprime(randInt(10000000000,100000000000000 )),randInt(10000000,10000000000 ),
randInt(10000000,10000000000 ),randInt(10000000,10000000000 ))
```

"Sie verabreden p "	35488283644651	"Sie verabreden g "	684331450
"Anton sendet "	31955541056774	"Berta sendet "	28253110606887
"Antons Schüssel "	10904195610372	"Bertas Schlüssel "	10904195610372

1.3

```
* onetimepad                                   8/14
Define LibPub onetimepad(mess,key)=
Func
© (message, key )–> (cryptogramm )
© alles als Ziffernlisten
Local dif,k
k:=key
Loop
dif:=dim(mess)–dim(k)
If dif=0 Then
Return mod(mess+k,10)
ElseIf dif<0 Then
k:=left(k,dim(k)+dif)
ElseIf dif>0 Then
k:=augment(k,k)
EndIf
EndLoop
EndFunc
```

onetimepad({2,5,0,3,4,8},{1,9,2,8,3,7}) ▸ {3,4,2,1,7,5}

Messagelänge=Schlüssellänge

onetimeinv({3,4,2,1,7,5},{1,9,2,8,3,7}) ▸ {2,5,0,3,4,8}

Wichtig für das Programm ist,

mod(*mess*+*k*,10)

Die Modulo–Funktion ist "listable", sie arbeitet auf den Elementen der Liste einzeln.

Ebenso kann man Listen elementweise mit + addieren.

{1,2,3}+{10,20,30} ▸ {11,22,33}

mod({11,22,33},7) ▸ {4,1,5}

Die anderen Programmteile dienen dazu, die Schlüssellänge an die Länge der Message anzupassen.

onetimepad({2,5,0,3,4,8},{1,9,2,8,3,7,4,6}) ▸ {3,4,2,1,7,5} Schlüssel länger

onetimeinv({3,4,2,1,7,5},{1,9,2,8,3,7,4,6}) ▸ {2,5,0,3,4,8}

onetimepad({2,5,0,3,4,8},{1,9,2}) ▸ {3,4,2,4,3,0} Schlüssel kürzer

onetimeinv({3,4,2,4,3,0},{1,9,2}) ▸ {2,5,0,3,4,8}

"onetimeinv" erfolgreich gespeichert

```
Define LibPub onetimeinv(cryp,key)=
Func
©(cryptogramm, key )->message
© alles als Ziffernlisten
Local dif,k
k:=key
Loop
dif:=dim(cryp)-dim(k)
If dif=0 Then
Return mod(cryp-k,10)
ElseIf dif<0 Then
k:=left(k,dim(k)+dif)
ElseIf dif>0 Then
k:=augment(k,k)
EndIf
EndLoop
EndFunc
```

onetimeinv({3,4,2,1,7,5},{1,9,2,8,3,7}) ▸ {2,5,0,3,4,8}

OneTimePad invers Kryptogramm und Schlüssel als Ziffernlisten.

onetimeinv({3,4,2,1,7,5},{1,9,2,8,3,7,4,6}) ▸ {2,5,0,3,4,8}

Schlüssel länger

onetimepad({2,5,0,3,4,8},{1,9,2}) ▸ {3,4,2,4,3,0}

Wenn der Schlüssel kürzer war, ist auch das Cryptogramm anders.

onetimeinv({3,4,2,4,3,0},{1,9,2}) ▸ {2,5,0,3,4,8}

1.5

zahl2li 6/9

```
Define LibPub zahl2li(zahl)=
Func
© zahl->Ziffernliste
Local i,z,li
li:={}
  z:=zahl
While z>0
li:=augment({mod(z,10)},li)
z:=floor(z/10)
EndWhile
Return li
EndFunc
```

```
zahl2li(112233445566) ▸ {1,1,2,2,3,3,4,4,5,5,6,6}
li2zahl({1,1,2,2,3,3,4,4,5,5,6,6}) ▸ 112233445566
```

1.6

li2zahl 0/8

```
Define LibPub li2zahl(liste)=
Func
© (Ziffernliste )–> Zahl
Local z,li,i
li:=liste: z:=0
For i,1,dim(liste)
z:=z·10+li[1]
li:=mid(li,2)
EndFor
Return z
EndFunc
```

```
li2zahl({2,5,0,3,4,8}) ▸ 250348
zahl2li(250348) ▸ {2,5,0,3,4,8}
```

1.7