Kryptofunktionen für TI Nspire Seite 1 /4

Stellen Sie die Datei **kry.tns** in das Verzeichnis MyLib in Ihrem TI-Nspire-Verzeichnis bei "Eigene Dateien". Dann können Sie mit kry*befehlsname* in jeder Datei die Befehle verwenden.

```
Define LibPriv ggte(a,b)=
Func
Local :r0,r1,r2,q0,q1,s0,s1,s2,t0,t1,t2:
r0:=a:r1:=b:
s0:=0:s1:=1:t0:=1:t1:=−ipart(a/b):
:loop
:   r2:=mod(r0,r1)
:   if r2=0 then
:      return [[r1,s0,t0]]
:      exit
:    endif
:    q0:=ipart(r0/r1)
:    q1:=ipart(r1/r2)
:    s2:=s0-q1*s1
:    t2:=t0-q1*t1
:    r0:=r1: r1:=r2: q0:=q1:
:    s0:=s1:s1:=s2:t0:=t1:t1:=t2
: endloop
:EndFunc
```

Erweiterter Euklidischer Algorithmus

Lokale Variable
Vorbelegungen

Schleife

Ausgabe des vorigen Restes, wenn der gerade erzeugte Rest 0 ist.
Ausgabe der vorigen Linearfaktoren.
Neuberechnung

ggte(13,8) [1,-3,5]
heißt 1=-3*13+3*8

MuPAD
igcdex(13,8) [1,-3,5]

gcd=greatest common divisor

```
Define LibPriv nextprime(a)=
Func
Local i,aa:
aa:=a:

If mod(aa,2)=0 Then
      i:=1:
   Else  i:=0:
EndIf:
Loop
   If isPrime(aa+i) Then
      Return aa+i:
:     Exit:
  EndIf:
  i:=i+2:
EndLoop:
EndFunc
```

Nächste Primzahl größer gleich a

Gerade Zahlen auslassen

Schleife von a an alle Ungeraden
eingebaute Funktion **IsPrime** nutzen
wenn das wahr ist, dann gib die Zahl aus

nächste Ungerade Zahl
wieder prüfen ...

120 Trillionen 31 ist die erste Primzahl hinter 120 Trillionen
Antwort in <1 Sekunden

nextprime(120000000000000)	120000000000031
factor(120000000000031·1−2)	1433849·83690821
isPrime(120000000000031)	true

120 Billlionen 31 ist die erste Primzahl hinter 120 Billionen
Antwort in <0,01 Sekunden

Kryptofunktionen für TI Nspire Seite 2/4

```
Define LibPriv zstern(a)=
Func
:Local i,s
:s:={1}
:For i,2,a-1
:  If gcd(a,i)=1 Then
:    s:=augment(s,{i})
:  EndIf
: EndFor
Return s:
:EndFunc
```

Prime Restklassen, $\mathbb{Z}_m^*$ wird bestimmt

Durchforsten, wenn ggT=1 dann in die
Liste s aufnehmen
Am Ende s ausgeben.

zstern(12) [1,5,7,11]

```
Define LibPriv euler(a)=
Func
Local i,s: s:=1:
For i,2,a-1
   If gcd(a,i)=1 Then
      s:=s+1:
    EndIf
EndFor
EndFunc
```

Eulersche Phi-Funktion
=Anzahl der Teilerfremden von a

Derselbe Algorithmus, nur wird s einfach hochgezählt.

MuPAD numlib::phi(a)

```
Define LibPriv eulerphi(m)=
Func
Return dim(zstern(m)):
EndFunc
```

Eulersche Phi-Funktion
als Anzahl der Elemente in $\mathbb{Z}_m^*$
$\varphi(m) = \mathbb{Z}_m^*$

```
Define LibPriv teiler(a)=
Func  Local i,t:   t:={}
:For i,1,floor(a)
:  If mod(a,i)=0 Then
:    t:=augment(t,{i})
:  EndIf:
:EndFor:
:Return t
:EndFunc
```

Teilermenge von a

$zstern(20)$	$\{1,3,7,9,11,13,17,19\}$
$euler(20)$	8
$eulerphi(20)$	8
$teiler(20)$	$\{1,2,4,5,10,20\}$

Kryptofunktionen für TI Nspire Seite 3/4

```
Define LibPriv pmod(a,k,m)=
Func:
        Local xx,kk,pot :
        kk:=k :xx:=1 :pot:=a :
        Loop :
            If mod(kk,2)=1 Then :
                xx:=mod(xx*pot,m) :
                If kk=1 Then :
                    Return xx :        Exit
:
                EndIf :      kk:=kk-1 :
            EndIf :    kk:=((kk)/(2)) :
            pot:=mod(pot*pot,m) :
        EndLoop:
EndFunc
```

PowerMod

MuPAD powermod(a,k,m)
dient dazu, die **Potenzierung im Modul** so geschickt zu machen, dass die zu hantierenden Zahlen nicht größer als das Quadrat des Moduls werden.
Die Idee basiert auf der "Doubbeldaddel"-Methode zur Erzeugung von Dualzahlen.
Die Hilfsvariable pot nimmt nacheinander die Werte an.

$$a, a^2, \left(a^2\right)^2, \left(\left(a^2\right)^2\right)^2, \ldots$$

Bei ungeraden Zwischenergebnis k_ wird das bisher Erreichte mit pot multipliziert, bei geradem nicht. Stets wird sofort der Rest modulo m gebildet. Ein ungerades k_ wird um 1 reduziert, ein gerades nicht. Das neue k_ ist die Hälfte davon.

```
Define LibPiv ordo(a,m)=
Func
:Local ordn,pot

:ordn:=1: pot:=mod(a,m):
:If gcd(a,m)>1 Then
:     Return "fail": Return
:EndIf:
:Loop
:   If pot=1 Then
:   Return ordn
:   Exit:
:   EndIf:
:   pot:=mod(pot*a,m):
:   ordn:=ordn+1:
:EndLoop:
:EndFunc
```

Ordnung eines Elementes

$\mathbb{Z}_m^*$ ist Gruppe.

Für Elemente a endlicher Gruppen ist die "**Ordnung von a**" der kleinste Exponent von a, der die Gruppe erzeugt.

Idee: einfach durchforsten, immer mehr a-Faktoren, wenn 1 herauskommt, wird der Exponent ausgegeben.

MuPAD numlib::order(a,m)

Kryptofunktionen für TI Nspire Seite 4/4

```
Define LibPriv maltafel(m)=
Func
:Local i,j,aa:
:aa:=newMat(m-1,m-1):
:For i,1,m-1
:For j,1,m-1
:  aa[i,j]:=mod(i*j,m):
:EndFor
:EndFor
:Return aa
:EndFunc

Define LibPriv malstern(m)=
Func
:Local i,j,aa,zs,k:
:zs:=zstern(m):
:k:=euler(m):
:aa:=newMat(k,k)
:For i,1,k
:For j,1,k
:  aa[i,j]:=mod(zs[i]*zs[j],m)
:EndFor
:EndFor
:Return aa
:EndFunc
Define LibPriv potstern(m)=
Func
:Local i,j,aa,zs,k
:zs:=zstern(m)
:k:=euler(m)
:aa:=newMat(k,k+1)
:For i,1,k
:For j,2,k+1
:  aa[i,j]:=pmod(zs[j-1],i,m):
:EndFor
:EndFor
:For i,1,k
:   aa[i,1]:=i:
:EndFor
:Return aa
:EndFunc
```

Multiplikationstafeln

Viel Arbeit erspart man sich, wenn man Multiplikationstafeln und Potenztafeln vom Rechner erstellen lässt.
Wichtig sind dabei folgende Befehle:

Erzeugen einer Matrix mit newMat(Zeilen,Sp)

Doppelte for-Schleife zum Belegen der Elemente.
diese spricht man mit Doppelindizes nach dem Namen an.

maltafel(6)

1	2	3	4	5
2	4	0	2	4
3	0	3	0	3
4	2	0	4	2
5	4	3	2	1

maltafel(5)

1	2	3	4
2	4	1	3
3	1	4	2
4	3	2	1

malstern(12)

1	5	7	11
5	1	11	7
7	11	1	5
11	7	5	1

potstern(18)

1	1	5	7	11	13	17
2	1	7	13	13	7	1
3	1	17	1	17	1	17
4	1	13	7	7	13	1
5	1	11	13	5	7	17
6	1	1	1	1	1	1

Bei den Maltafeln sind Eingangszeile und ~Spalte weggelassen, da sich auch so klar zu sehen sind.
Bei den Potenztafeln stehen in der ersten Spalte die Exponenten
In der ersten Zeile sind ab Platz 2 die Elemente von $\mathbb{Z}_m^*$ zu sehen.
Darunter stehen alle Potenzen dieser Elemente.
Die erste 1 pro Spalte ergibt am Zeilenanfang die Ordnung eines Elements. ord(7)=3