Quelltext der einzelnen Programme:

Transposition:

Verschlüsselung:

```
:transp()
:Prgm
:ClrIO
:
:Local code,l,s,klt
:InputStr "Text:",klt
:Input "Schlüssel: ",s
:dim(klt)→l
:
:Loop
: If mod(l,s)≠0 Then
:  klt&"x"→klt
:
:  dim(klt)→l
:
: Else
:  Goto a1
: EndIf
:EndLoop
:
:Lbl a1
:newList(l)→code
:
:For i,1,l,1
: mid(klt,i,1)→code[i]
:EndFor
:
:
:list▸mat(code,s)→code
:mat▸list(code^T)→code
:""→out
:
:For j,1,l,1
: out&code[j]→out
:EndFor
:
:Disp "Geheimtext: ",out
:EndPrgm
```

Eingabe des Textes und des Schlüssels

Auffüllen aller Felder

Initialisieren einer neuen Liste.
ASCII Code wird herausgelesen und auf Listenelemente abgespeichert
Liste wird in Matrix umgewandelt
Transponierte Matrix wird in Liste zurückgewandelt
Geheimtext wird zusammengesetzt und angezeigt

Entschlüsselung:

```
:enttrans(out)
:Prgm
:ClrIO
:
:Local code,l,s
:Disp "Geheimtext:",out
:Input "Schlüssel: ",s
:dim(out)→l
:Loop
: If mod(l,s)≠0 Then
:  out&"x"→out
:  dim(out)→l
:
: Else
:
:  Goto a1
: EndIf
:EndLoop
:
:Lbl a1
:newList(l)→code
:
:For i,1,l,1
: ord(mid(out,i,1))→code[i]
:
:EndFor
:
:list▸mat(code,l/s)→teile
:mat▸list(teile^T)→teile
:""→ght
:For j,1,l,1
:ght&char(teile[j])→ght
:EndFor
:
:Disp "Originaltext: ",ght
:EndPrgm
```

Verschlüsselte Text (Chiffretext) wird an das Programm übergeben und angezeigt.
Schlüssel wird eingegeben.

Felder werden wieder aufgefüllt
(Nur nötig, wenn ein verschlüsselter Text eingegeben wird. Der übergebene Chiffretext hat bereits die gewünschte Länge)

entschlüsseln

Entschlüsselter Text wird ausgegeben.

Cäsar-Code

Verschlüsselung:

```
:vercaes()
:Prgm
:ClrIO
:Local s,klar,l,code
:
:InputStr "Text: ",klar
:Input "Verschiebung ",s
:dim(klar)→l
:
:""→out
:For i,1,l,1
:ord(mid(klar,i,1))→code
:
:If code>64 and code<91 Then
: 65+mod(code-65+s,26)→code
:ElseIf code>96 and code<123 Then
: 97+mod(code-97+s,26)→code
:Else
: code→code
:EndIf
:out&char(code)→out
:EndFor
:Disp out
:EndPrgm
```

Eingabe des Textes und des Schlüssels (= Verschiebung)

Großbuchstaben

Kleinbuchstaben

Leerzeichen, Sonderzeichen,…

Ausgabe Geheimtext

Entschlüsselung

```
:entcaes(klar)
:Prgm
:ClrIO
:Local r,s,klar,l,ght,code
:Disp "Text: ",klar
:Input "Verschiebung ",s
:
:dim(klar)→l
:
:""→ght
:For i,1,l,1
:ord(mid(klar,i,1))→code
:
:If code>64 and code<91 Then
: 65+mod(code-65-s,26)→code
:ElseIf code>96 and code<123 Then
: 97+mod(code-97-s,26)→code
:Else
: code→code
:EndIf
:ght&char(code)→ght
:EndFor
:Disp ght
:EndPrgm
```

Geheimtext wird an Programm übergeben und angezeigt.
Schlüssel wird eingegeben.

Großbuchstaben

Kleinbuchstaben

Leerzeichen, Sonderzeichen,…

Ausgabe Klartext

Vignere-Code:

Verschlüsselung

```
:vign()
:Prgm
:ClrIO
:Local ght,cod,l,a,r,z,j,r,kl
:InputStr "Klartext:",ght
:
:InputStr "Gib das Codewort ein:",cod
:dim(ght)→k
:dim(cod)→l
:newList(l)→code
:
:For i,1,l,1
: ord(mid(cod,i,1))→code[i]
:EndFor
:
:newList(k)→klt
:For i,1,k,1
: ord(mid(ght,i,1))→klt[i]
:EndFor
:
:newList(k)→trans
:1→z
:1→j
:For i,1,l,1
:
: 0→r
: While j<k+1
:  code[i]→trans[j]
:  r+1→r
:  i+r*l→j
: EndWhile
: z+1→z
: z→j
:EndFor
:
:For i,1,k,1
: If klt[i]>64 and klt[i]<91 and trans[i
 ]>96 and trans[i]<123 Then
:  65+mod(trans[i]+klt[i]-162,26)→klt[i]
:
:  ElseIf trans[i]>64 and trans[i]<91 an
 d klt[i]>64 and klt[i]<91 Then
:  65+mod(klt[i]+trans[i]-130,26)→klt[i]
:
:  ElseIf klt[i]>96 and klt[i]<123 and t
 rans[i]>64 and trans[i]<91 Then
:  97+mod(klt[i]+trans[i]-162,26)→klt[i]
:
:  ElseIf trans[i]>96 and trans[i]<123 a
 nd klt[i]>96 and klt[i]<123 Then
:  97+mod(klt[i]+trans[i]-194,26)→klt[i]
:
:  Else
:  klt[i]→klt[i]
: EndIf
:EndFor
:
:""→out
:
:For i,1,k,1
: out&char(klt[i])→out
:EndFor
:Disp out
:
:EndPrgm
```

Eingabe des Geheimtextes und des Codewortes

Codewort wird in Liste umgewandelt

Geheimtext wird in Liste umgewandelt

Codewort wird immer wieder in eine Liste geschrieben, die genau so lang ist wie der Geheimtext.

Zum Geheimtext wird der passende Buchstabe des Codewortes addiert. 4 Fälle (Groß- und Kleinschreibung) werden unterschieden, wobei die Schreibweise des Klartextes unverändert bleibt..

Sonderzeichen und Blank bleiben unverschlüsselt

Verschlüsselte Text wird ausgegeben.

Entschlüsselung:

```
F1 F2 Control F3 I/O F4 Var F5 Find... F6 Mode
:entvign(ght)
:Prgm
:ClrIO
:Local cod,ght,k,l,a,r,code,klt,trans
:Disp "Geheimtext: ",ght
:InputStr "Gib das Codewort ein:",cod
:dim(ght)→k
:dim(cod)→l
:int(k/l)→a
:mod(k,l)→r
:newList(l)→code
:
:For i,1,l,1
: ord(mid(cod,i,1))→code[i]
:EndFor
:
:newList(k)→klt
:For i,1,k,1
: ord(mid(ght,i,1))→klt[i]
:EndFor
:
:newList(k)→trans
:1→z

:1→j
:For i,1,l,1
: 0→r
: While j<k+1
:  code[i]→trans[j]
:  r+1→r
:  i+r*l→j
: EndWhile
: z+1→z
: z→j
:EndFor

:For i,1,k,1
:If klt[i]>64 and klt[i]<91 and 32+klt[i
 ]>trans[i] Then
:  65+mod(klt[i]-trans[i]+32,26)→klt[i]
: ElseIf klt[i]+32≤trans[i] and klt[i]>6
 4 and klt[i]<91 Then
:  65+mod(26+klt[i]+32-trans[i],26)→klt[
 i]
:ElseIf klt[i]>96 and klt[i]<123 and klt
 [i]>trans[i] Then
:  97+mod(klt[i]-trans[i],26)→klt[i]
:
: ElseIf klt[i]≤trans[i] and klt[i]>96 a
 nd klt[i]<123 Then
:  97+mod(26+klt[i]-trans[i],26)→klt[i]
:
: Else
: klt[i]→klt[i]
:EndIf
:EndFor
:""→ght
:For i,1,k,1
: ght&char(klt[i])→ght
:
: ElseIf klt[i]≤trans[i] and klt[i]>96 a
 nd klt[i]<123 Then
:  97+mod(26+klt[i]-trans[i],26)→klt[i]
:
: Else
: klt[i]→klt[i]
:EndIf
:EndFor
:""→ght
:For i,1,k,1
: ght&char(klt[i])→ght
:
:EndFor
:Disp ght
:EndPrgm
MAIN DEG AUTO FUNC
```

Eingabe des Codewortes

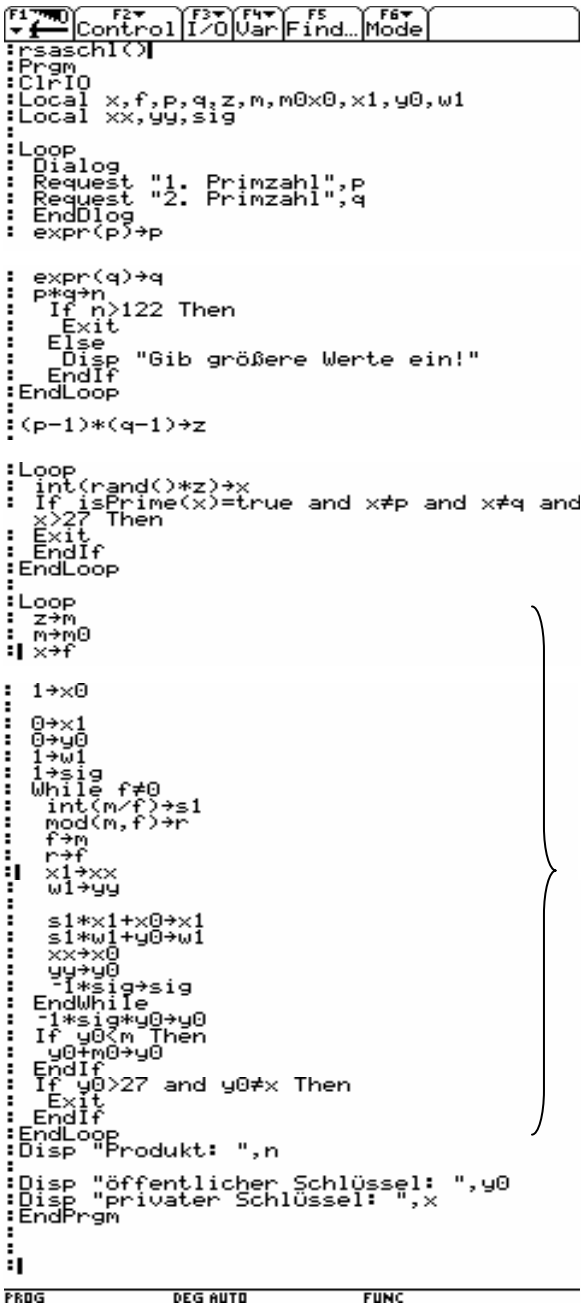
Siehe Verschlüsselung

Geheimtext wird mit dem Codewort entschlüsselt. (Achtung auf „Überlauf“ bei modularer Rechnung!)

Zusammensetzen und anzeigen des Klartextes

RSA-Verfahern:

Erzeugung der Schlüssel:



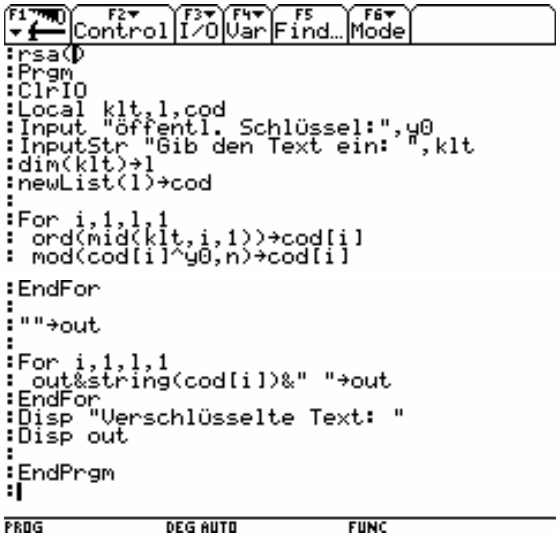
Eingabe der beiden Primzahlen; Produkt darf nicht kleiner als 122 sein! (max. Wert bei ASCII-Code)
Formatumwandlung
Berechnung des Produkts und der Eulerschen Zahl

Bestimmung des privaten Schlüssels mittels Zufallszahl

Euklidscher Algorithmus zur Bestimmung des öffentlichen Schlüssels

Anzeige des öffentlichen, des privaten Schlüssels und des Produktes

Verschlüsselung:



```
:rsa()
:Prgm
:ClrIO
:Local klt,l,cod
:Input "öffentl. Schlüssel:",y0
:InputStr "Gib den Text ein: ",klt
:dim(klt)→l
:newList(l)→cod
:
:For i,1,l,1
: ord(mid(klt,i,1))→cod[i]
: mod(cod[i]^y0,n)→cod[i]
:EndFor
:
:""→out
:
:For i,1,l,1
: out&string(cod[i])&" "→out
:EndFor
:Disp "Verschlüsselte Text: "
:Disp out
:
:EndPrgm
:
```

Eingabe des öffentlichen Schlüssels und des Klartextes

Umwandlung in ASCII – Code und Verschlüsselung

Zusammensetzen und anzeigen des verschlüsselten Textes

Entschlüsselung

```
:entrsa(ght)
:Prgm
:ClrIO
:Local i,j,zw,zw1,zw2,ght,x,l,code
:Disp "Geheimtext: ",ght
:Input "privater Schlüssel:",x
:
:dim(ght)→l
:""→klar
:newList(l)→code
:For i,1,l,1
:mid(ght,i,1)→code[i]
:EndFor
:newList(l)→zw2
:1→j
:1→i
:While i<l+1
: ""→zw
: Loop
:  While i<l+1
:   If ord(code[i])≠32 Then
:    zw&code[i]→zw
:    i+1→i
:   Else
:    Exit
:   EndIf
:  EndWhile
:  Exit
: EndLoop
:
: expr(zw)→zw1
: mod(zw1^x,n)→zw2[j]
: j+1→j
: i+1→i
:
:EndWhile
:""→klar
:For i,1,j,1
:klar&char(zw2[i])→klar
:EndFor
:
:Disp "Klartext: "
:Disp klar
:EndPrgm
```

Eingabe des privaten Schlüssels und des verschlüsselten Textes.

Abspeichern jeder Zahl des Geheimtextes in einer Liste

Herauslesen der Zahlen aus der Liste, die zwischen zwei Leerzeichen stehen und die ursprüngliche Zahl wird aus diesen zusammengesetzt.

Entschlüsseln des Textes

Zusammensetzen und anzeigen des entschlüsselten Textes.