WQ, FS 2013

RSA -- Verschlüsselung , Buchstaben werden numeriert :

```
  (*  leer= 00,  A= 01,  B=02,   C=03, ... u.s.w. *)
```

je 3 Zeichen werden zusammengefasst zu einem "Wort" ;
das grösste Wort ist damit ZZZ = 262 626

(1) Wähle p = 307, q = 859, bilde n = p * q

```
p = 307; q = 859;  n = 307 * 859
```

263 713

```
PrimeQ[p]
PrimeQ[q]
```

True

True

```
(* Somit ist die EulerFunktion φ *)
φ[n_] = EulerPhi[n]
```

262 548

```
(* Diese ist hier sehr leicht berechenbar mit (p-1)*(q-1) *)
(p - 1) * (q - 1) == φ[n]
```

True

```
(* Man bemerke:  p,q sind so gewählt,
dass n > 262626 ist -- eine notwendige Bedingung  *)
```

(2) Suche einen öffentlichen Schlüssel ee teilerfremd zu φ[n],
wähle willkuerlich eine nicht zu kleine Primzahl

```
ee = 1721;
(* Test *) GCD[262 548, ee]
```

1

```
(* Test *) PrimeQ[1721]
```

True

```
(* Erweiterter Euklidischer Algorithmus fuer den geheimen Schlüssel dd *)
 ExtendedGCD[262 548, ee]
```

{1, {-9, 1373}}

```
(* Es ist also *) 1 == -9 * 262548 + 1373 * 1721
```

```
True
```

```
dd = 1373;
```

```
(* direkt geht diese Berechnung auch mit *)
```

```
dd = PowerMod[1721 , -1, 262548]
```

```
1373
```

```
(* PowerMod berechnet die inverse Restklasse zu 1721 hier modulo φ[n] *)
```

(3) Der zu verschlüsselnde (sinnfreie) Text sei :
   Q U A P P W A R K L A S S E

```
text = {172101, 161600, 230118, 001112, 011919, 050000 }
```

```
{172101, 161600, 230118, 1112, 11919, 50000}
```

```
(* Verschlüsselung   W_j=K_j^ee mod (n)  ,  {j,1,6}   *)
```

```
sende = Mod[ text^ee , n]
(*  hier koennen bei W_1 und W_6 noch fuehrende Nullen ergaenzt werden;
da es aber nur um Zahlen geht,   kann das unterbleiben *)
```

```
{14399, 243824, 254058, 104736, 128177, 71246}
```

(4) Entschlüsselung mit dd : K_j = W_j^dd mod (n)

```
neuerText = Mod[ sende^dd , n]
```

```
{172101, 161600, 230118, 1112, 11919, 50000}
```

```
(*  K_4 und K_6 müssen nun mit führenden Nullen aufgefüllt werden. Die
 Rückübersetzung der Zahlen in die Buchstaben ergibt den Ausgangstext.

  Die Rückübersetzung funktioniert wegen des kleinen
 Satzes von Fermat: Wenn x und n teilerfremd sind, gilt
   x^φ[n] ≡ 1 mod n
    Wir haben
    1 + 9*262548 = 1373*1721 , allgemein 1 + k φ[n] = ee dd ,
x sei der Text, dann erhaelt man nach Ver- und Entschluesselung
   (x^ee)^dd = x^(1 + k φ[n])=x*(x^φ[n])^k ≡ x mod n  *)
```

```
(*  Anhang Extra-Uebung: Euklidischer Algorithmus by hand *)
```

```
eukal[a0_Integer, b0_Integer] :=
 Module[{a = a0, b = b0, q, r},
  While[b ≠ 0, r = Mod[a, b]; q = (a - r) / b;
   Print[a, "=", q, "*", b, "*", r];
   {a, b} = {b, r}]; a]
```

```
eukal[262548, 1721]
```

```
262 548=152*1721*956

1721=1*956*765

956=1*765*191

765=4*191*1

191=191*1*0

1
```

```
(* Anhang Uebung: Erweiterter Euklidischer Algorithmus by hand *)
```

In[1]:=
```
extendedEukl[a0_Integer, b0_Integer] :=
 Module[{a = a0, b = b0, q, r, ua = 1, ub = 0, va = 0, vb = 1},
  While[b != 0,
   r = Mod[a, b]; q = (a - r) / b;
   {a, b} = {b, r};
   {ua, ub} = {ub, ua - q * ub};
   {va, vb} = {vb, va - q * vb};
   Print[r, " = (", ub, ")*", a0, "+(", vb, ")*", b0, " = ", ub * a0 + vb * b0]
  ]; {a, ua, va}]
```

In[2]:=
```
extendedEukl[262 548, 1721]
```

```
956 = (1)*262 548+(-152)*1721 = 956

765 = (-1)*262 548+(153)*1721 = 765

191 = (2)*262 548+(-305)*1721 = 191

1 = (-9)*262 548+(1373)*1721 = 1

0 = (1721)*262 548+(-262 548)*1721 = 0
```

Out[2]=
```
{1, -9, 1373}
```

```
(* Anhang zu Rundungsfehler in Ueb14.  Es ist s^2=6  *)

s = Sqrt[2 + Sqrt[3]] + Sqrt[2 - Sqrt[3]] ≤ Sqrt[6]
s // N
```

N::meprec : Internal precision limit $MaxExtraPrecision

= 50.` reached while evaluating $-\sqrt{6} + \sqrt{2-\sqrt{3}} + \sqrt{2+\sqrt{3}}$. ≫

$$\sqrt{2-\sqrt{3}} + \sqrt{2+\sqrt{3}} \le \sqrt{6}$$

```
True

s // RootReduce

True

s[[1]] - s[[2]] // N
```

2.22045×10^{-16}

```
s[[1]] == s[[2]] // N
```

```
True
```

```
s[[1]] - s[[2]] == 0 // N
```

```
False
```

```
(s[[1]] - s[[2]] // N // Chop ) == 0
```

```
True
```