



Aufgabe 1. Lade dir Cygwin herunter und installiere das Paket für GMP. Kompiliere folgendes Beispielprogramm

```
1 /* Gebe einige Zufallszahlen aus. */
2 #include <stdio.h>
3 #include <gmp.h>
4 #include <time.h>
5
6 int main() {
7     mpz_t a;
8     mpz_init(a);
9     mpz_set_str(a, "856490000", 10);
10    mpz_mul(a, a, a);
11    gmp_printf("%Z\n", a);
12    mpz_clear(a);
13    return 0;
14 }
```

In den folgenden Aufgaben geht es um elliptische Kurven. Zum Testen empfehlen wir

$$E := \{ (x, y) \in \mathbb{F}_p^2 \mid y^2 = x^3 + ax + b \} \cup \{ \mathcal{O} \}$$

mit den folgenden Parametern aus dem Brainpool Standard ¹:

```
1 Curve-ID: brainpoolP256r1
2 p = A9FB57DBA1EEA9BC3E660A909D838D726E3BF623D526202820
3     13481D1F6E5377
4 a = 7D5A0975FC2C3057EEF67530417AFFE7FB8055C126DC5C6CE9
5     4A4B44F330B5D9
6 b = 26DC5C6CE94A4B44F330B5D9BBD77CBF958416295CF7E1CE6B
7     CCDC18FF8C07B6
8 x = 8BD2AEB9CB7E57CB2C4B482FFC81B7AFB9DE27E1E3BD23C23A
9     4453BD9ACE3262
10 y = 547EF835C3DAC4FD97F8461A14611DC9C27745132DED8E545C
11     1D54C72F046997
12 q = A9FB57DBA1EEA9BC3E660A909D838D718C397AA3B561A6F790
13     1E0E82974856A7
14 h = 1
```

¹<http://www.ecc-brainpool.org/download/draft-lochter-pkix-brainpool-ecc-00.txt>



Die Parameter sind im Hexadezimalsystem angegeben. Hier ist außerdem $g = (x, y)$ ein Punkt auf der Kurve und $q = |\langle g \rangle|$ die Anzahl Punkte in der von g erzeugten Untergruppe.

Aufgabe 2. Implementiere unter Verwendung der GMP ein Modul, welches die Punktgruppe einer elliptischen Kurve implementiert. Es sollten mindestens Funktionen zur Verfügung stehen, Punkte auf einer durch Parameter a und b gegebenen Kurve zu addieren.

Aufgabe 3. Implementiere eine Funktion, die zu einem Punkt $p \in E$ und einer Zahl $n \in \mathbb{Z}_+$ den Punkt $n \cdot p \in E$ auf der Kurve berechnet. Eine *naive* Möglichkeit ist es selbstverständlich, den Punkt p genau n mal aufzuaddieren. Wir kennen jedoch eine Methode zur schnellen Potenzierung von Zahlen, die prinzipiell auch hier anwendbar ist. (Stichwort: Double & Add)

Aufgabe 4. Implementiere Lenstras EC-Faktorisierungsalgorithmus. Eine Zahl zum testen wird auf der Kurshomepage zum Download verfügbar sein.

Algorithmus 1 Lenstras EC-Faktorisierungsalgorithmus

Input: Zahl $n \in \mathbb{Z}_+$.

Output: Ein Teiler d von n

```
1: Wähle zufällige  $x_0, y_0, a \in \mathbb{Z}_n$ 
2: set  $b := y_0^2 - x_0^3 - ax_0$ 
3: Sei  $E$  die Kurve zu den Parametern  $a$  und  $b$ .
4: set  $p := (x_0, y_0) \in E$ 
5: for  $m = 2, 3, 4, \dots$  do
6:   set  $p := m \cdot p \pmod{n}$ 
7:   if Berechnung schlägt fehl then
8:     Wir haben ein  $x$  gefunden, welches kein Inverses modulo  $n$  hat.
9:     set  $d := \gcd(x, n)$ 
10:    if  $d < n$  then
11:      return  $d$ 
12:    else if  $d = n$  then
13:      Starte Algorithmus neu
14:    end if
15:  end if
16: end for
```
