– Praktikumsaufgabe 6 –

Thema: *Langzahlarithmetik*

Zielstellung: Kennenlernen einer Bibliothek zur Arithmetik mit langen Zahlen

1. Machen Sie sich mit der `libgmp` vertraut und installieren Sie diese lokal in Ihrem Homeverzeichnis!

   a) Laden Sie das Quellarchiv von *http://www.gmplib.org* herunter.

   b) Beweisen Sie die Integrität des heruntergeladenen Archivs, indem Sie die kryptografischen Hashes prüfen und die angegebene Signatur verifizieren.

   c) Konfigurieren und übersetzen Sie die Bibliothek und testen Sie deren Funktionsfähigkeit.

   Hinweise:

   - Da Sie in den Pools keine Root-Rechte besitzen, dürfen Sie keine Bibliotheken in das System einspielen. Sie können aber jederzeit Bibliotheken in Ihr eigenes Homeverzeichnis[1] installieren.

   - Die einzelnen Befehlsschritte sind die folgenden:

     a) das finale Installationsverzeichnis anlegen

     ```
     robge@ilux150:~> mkdir ~/libgmp
     ```

     b) das Quellarchiv der Bibliothek an einen geeigneten Platz (im Beispiel das Verzeichnis `~/src`) herunterladen

     ```
     robge@ilux150:~> cd src
     robge@ilux150:~/src> wget https://gmplib.org/download/gmp/gmp
        -6.2.1.tar.lz
     ```

     c) Das Archiv entpacken

     ```
     robge@ilux150:~/src> tar xf gmp-6.2.1.tar.lz
     ```

     d) Die Quellen konfigurieren, dabei das finale Installationsverzeichnis festlegen (es muss ein *absoluter* Pfad angegeben werden, den Sie über das Kommando `pwd` 'rauskriegen)

     ```
     robge@ilux150:~/src/gmp-6.2.1> ./configure --prefix /user/
        profile/active/s12345/libgmp
     ```

     e) die Bibliothek übersetzen

[1] Vorausgesetzt, Sie haben genügend Platz. Sie benötigen etwa 50 MiB.

```
robge@ilux150:~/src/gmp-6.2.1> make
```

f) Die übersetzte Bibliothek auf Korrektheit überprüfen:

```
robge@ilux150:~/src/gmp-6.2.1> make check
```

g) und schließlich (in das *Installationsverzeichnis*) installieren

```
robge@ilux150:~/src/gmp-6.2.1> make install
```

h) Die Dokumentation erzeugen (und lesen!):

```
robge@ilux150:~/src/gmp-6.2.1> cd doc
robge@ilux150:~/src/gmp-6.2.1/doc> make gmp.pdf
```

- Wenn Sie Bibliotheken (wie in diesem Falle) in nichtstandardisierte Verzeichnisse einspielen, dann müssen Sie

  a) den Compiler über den Ort der Headerdatei informieren,

  b) den Linker über den Ort der Bibliothek informieren,

  c) den Loader ebenfalls über den Ort der Bibliothek informieren, sofern Sie nicht statisch linken.

  Dies geschieht mittels der folgenden 3 Switches, die Sie beim Übersetzen angeben müssen (`<instpath>` muss durch das konkrete Installationsverzeichnis ersetzt werden):

  ```
  -I<instpath>/include
  -L<instpath>/lib64
  -Wl,--rpath -Wl,<instpath>/lib64
  ```

  Zusätzlich müssen Sie dem Linker mitteilen, welche Bibliothek gelinkt werden soll (`-lgmp`). Zur Vereinfachung stelle ich ein `Makefile` zur Verfügung, bei dem Sie nur die Variablen `P` (der Name des zu übersetzenden Programmes) und `INSTALL` (das Installationsverzeichnis der Bibliothek) anpassen müssen.

2. Prüfen Sie, ob die im Zuge der *RSA Factoring Challenge* ermittelten Primfaktoren der Zahl RSA-100 wirklich stimmen (durch Multiplikation derselben)!

3. Finden Sie eine mindestens 30stellige Primzahl!

4.* a) Versuchen Sie, die Zahl 1524157897241274137 in ihre Primfaktoren zu zerlegen, nutzen Sie die probeweise Division!

   b) Versuchen Sie, die folgende (Dezimal-)Zahl n zu faktorisieren.

   $$n = 37858401860977862723111204717599037173669147721853067190597$$

   Nutzen Sie den Algorithmus von Fermat!

5.* Implementieren Sie das *Sieb des Eratosthenes* (ohne Nutzung der `libgmp`) und versuchen Sie, eine möglichst große Primzahl damit zu generieren.

6. Gegeben sei die folgende RSA-verschlüsselte Symbolfolge:

   669, 1313, 745, 1773, 205, 3025, 2570, 1313, 2412, 2923, 1632, 2570, 1313, 436, 1992, 1802, 624, 1292, 2816, 529, 1794, 1956, 538, 624, 180, 678, 1992, 2906, 538, 1292, 2816, 368, 1794, 1956, 368, 538, 180, 0

   sowie der *öffentliche* Schlüssel $(e, n) = (17, 3233)$.

   Ermitteln Sie die Klartextnachricht unter der Voraussetzung, dass jedes Byte einzeln verschlüsselt wurde und kein Padding zum Einsatz kam. Können Sie mit dem Klartext etwas anfangen?

   Hinweise:

   - Die Entschlüsselung erfolgt mit dem privaten Schlüssel, den Sie irgendwie herauskriegen müssen.
   - Die Funktionen `mpz_divisible()`, `mpz_divexact()` könnten nützlich sein, um den Schlüssel zu bestimmen.
   - Anstelle den erweiterten euklidischen Algorithmus zu implementieren, können Sie die Funktion `mpz_invert()` nutzen, in der dies bereits verwirklicht ist.
   - Zum Entschlüsseln benötigen Sie ggf. noch die Funktion `mpz_powm()`.

7. Ermitteln Sie die Werte der folgenden arithmetischen Ausdrücke ohne Einsatz eines Rechners (Übung zur Klausur!). Geben Sie jeweils alle Zwischenergebnisse sowie Ihren Rechenweg an und vermeiden Sie Zwischenergebnisse größer als 255.

   a) $83 \cdot 97 \bmod 11$

   b) $\varphi(143)$

   c) $4^{13} \bmod 13$

8. Gegeben seien $p = 3, q = 11, d = 7$ und $x = 5$. Verschlüsseln Sie x und ermitteln Sie das dafür nötige e nur unter Nutzung von Papier, Stift und eines (nichtprogrammierbaren) Taschenrechners.

9. Generieren Sie alle Carmichael-Zahlen im Intervall [1; 10.000.000].