Hochschule für Technik und Wirtschaft
Studiengang Kommunikationsinformatik
Studiengang Praktische Informatik
Prof. Dr.–Ing. Damian Weber

Kryptographie – Übung 2

Aufgabe 1 (Rechnen in $\mathbf{Z}/n\mathbf{Z}$)

Implementieren Sie eine Klasse `znz`, die das Rechnen modulo n vereinfacht; diese Klasse sollte für alle Instanzen das gleiche n verwenden. Es sollte so sein, daß man dieses n nur einmal, und zwar vor Beginn aller Berechnungen, setzen muß.

Als Konstruktoren sollten Sie die Initialisierung eines `znz` Elements wahlweise mit einem `int`, `BigInteger` und einem `String` vorsehen.

Als Methoden sollten die Grundrechenarten und das Potenzieren möglich sein.

Schreiben Sie einen kleinen Interpreter, der die Grundrechenarten mod n unterstützt, z.B. Rechnen mod 990366689399:

```
$ java interpreter 990366689399
>111111111111+888888888888
9633310600
>111111111111-888888888888
212588911622
>111111111111*888888888888
127922799802
>111111111111/888888888888
123795836175
>111111111111^888888888888
223751932320
>quit
$
```

Hierzu müssen Sie für die Klasse `znz` die Methoden, die die Arithmetik ausführen analog zu `BigInteger` entwickeln, d.h. die Namensgebung und die Bedeutung der Methoden soll für die beiden Klassen die gleiche sein.

Was kommt für $a^{990366689399}$ mod 990366689399 für beliebige a heraus?

Aufgabe 2 (Primzahlen)

a) Implementieren Sie den Fermat'schen Primzahltest und seine Verbesserung für den Exponent $\frac{p-1}{2}$.

b) Implementieren Sie den Miller–Rabin–Primzahltest. Nehmen Sie p als Primzahl an, wenn der Miller–Rabin–Primzahltest für alle $a \in \{2, 3, 5, 7, 11\}$ funktioniert hat.

c) Finden Sie alle Zahlen aus $\{11, 13, 15, \ldots, 10^7 - 1\}$, die laut b) nicht prim sind, aber den (nicht verbesserten) Fermat'schen Primzahltest aus a) für $a = 2$ bestehen.

d) Implementieren Sie eine Funktion

`BigInteger nextprime(BigInteger n)`

die für eine gegebene Zahl die nächste Primzahl $p > n$ zurückliefert.

Hierfür können Sie als Basis Ihre Miller–Rabin–Methode benutzen.

Bemerkung: `boolean isProbablePrime(int)` ist die in der JAVA–Klassenbibliothek enthaltene Miller–Rabin–Implementierung.