

# Kapitel 6

## Zahlentheorie

### Verständnisfragen

#### Sachfragen

1. Erläutern Sie die ganzzahlige Division!
2. Was versteht man unter  $a \bmod b$  für  $a, b \in \mathbb{Z}$ ?
3. Erläutern sie  $a|b$  für  $a, b \in \mathbb{Z}$ !
4. Was ist eine Primzahl?
5. Was ist eine zusammengesetzte Zahl?
6. Wie viele Primzahlen gibt es?
7. Wie lautet der Fundamentalsatz der Zahlentheorie?
8. Kennen Sie einen einfachen Primzahltest?
9. Erläutern Sie das Sieb des Erathostenes!
10. Erläutern Sie die Funktion  $\pi(n)$ !
11. Wie lautet die Definition des größten gemeinsamen Teilers?
12. Welche Berechnungsmöglichkeiten für den größten gemeinsamen Teiler kennen Sie?
13. Was ist eine diophantische Gleichung?
14. Wie verläuft der Euklidische Algorithmus?
15. Wie verläuft der verallgemeinerte Euklidische Algorithmus?
16. Wie kann der größte gemeinsame Teiler rekursiv berechnet werden?
17. Was sind zwei ganze Zahlen kongruent zum Modul  $m$ ?
18. Erläutern Sie die modulare Addition und Multiplikation!
19. Erläutern Sie den Zusammenhang zwischen Rest- und Äquivalenzklassen!
20. Wie lautet die Kürzungsregel für die modulare Division?

21. Was ist eine modulare Inverse? Wie kann sie bestimmt werden?
22. Beschreiben Sie den Graphen der Funktion  $n^2 \bmod m$ !
23. Erläutern Sie die modulare Wurzel und den modularen Logarithmus!
24. Wie lautet der kleine Satz von Fermat?
25. Wie ist die Euler'sche Funktion definiert?
26. Wie kann die Euler'sche Funktion berechnet werden?
27. Wie lautet der Satz von Euler?
28. Erläutern Sie die Aufgabenstellung der Kryptographie!
29. Erläutern Sie die Verschiebe-Chiffre!
30. Erläutern Sie die Tausch-Chiffre!
31. Was muss der Schlüssel einer Tausch-Chiffre erfüllen?
32. Erläutern Sie das Prinzip eines Public-Key-Kryptosystems!
33. Erläutern Sie das Kryptosystem nach Hellman, Merkle, Diffie!
34. Erläutern Sie das RSA-System!
35. Erläutern Sie die digitale Unterschrift mit Hilfe des RSA-Systems!
36. Erläutern Sie das ElGamal-System!

### Methodenfragen

1. Den Quotienten und den Rest bei der ganzzahligen Division angeben können.
2. Teiler einer Zahl angeben können.
3. Die Regeln für die Teilbarkeit anwenden können.
4. Primfaktoren einer Zahl finden können.
5. Die Primfaktorzerlegung für kleine Zahlen konstruieren können.
6. Das Sieb des Erathostenes mit Papier und Bleistift durchführen können.
7. Das Sieb des Erathostenes programmieren können.
8. Den Euklidischen Algorithmus durchführen können.
9. Den verallgemeinerten Euklidischen Algorithmus durchführen können.
10. Einfache diophantische Gleichungen lösen können.
11. Den größten gemeinsamen Teiler zweier Zahlen rekursiv berechnen können.
12. Den Euklidischen Algorithmus und seine Verallgemeinerung programmieren können.
13. Die Funktion  $\pi(n)$  mit Hilfe eines Siebs berechnen können.
14. Kongruenzen aufstellen und nachrechnen können.
15. Modulare Arithmetik ausführen können; insbesondere die Kürzungsregel der modularen Division.

16. Die modulare Inverse berechnen können und zur Gleichungslösung verwenden können.
17. Modulare Potenzrechnung, Wurzeln und Logarithmen bestimmen können.
18. Einen Primzahltest auf der Basis des kleinen Satzes von Fermat durchführen können.
19. Die Euler'sche Funktion berechnen können.
20. Eine Verschiebe-Chiffre anwenden und programmieren können.
21. Entscheiden können, ob ein gegebenes Zahlenpaar als Schlüssel einer Tausch-Chiffre verwendet werden kann.
22. Eine Tausch-Chiffre anwenden und programmieren können.
23. Das Krypto-System nach Hellman, Merkle, Diffie anwenden und programmieren können.
24. Das RSA-System anwenden und programmieren können.
25. Das ElGamal-System anwenden und programmieren können.

## Übungsaufgaben

1. Geben Sie den Quotienten  $q$  und den Rest  $r$  für die folgenden Zahlenpaare an:  $a = 7, b = 5$ ;  $a = 32, b = 11$ ,  $a = -32, b = 11$ ,  $a = 625, b = 24$  und  $a = 774, b = 354$ .

*Lösung:*

$$7 = 1 \cdot 5 + 2, \text{ also } q = 1, r = 2.$$

$$32 = 2 \cdot 11 + 10, \text{ also } q = 2, r = 10.$$

$$-32 = (-3) \cdot 11 + 1, \text{ also } q = -3, r = 1.$$

$$625 = 26 \cdot 24 + 1, \text{ also } q = 26, r = 1.$$

$$774 = 2 \cdot 354 + 76, \text{ also } q = 2, r = 76.$$

2. Bestimmen Sie die Primzahlfaktorisation von 1 001, 7 777 und 10 121 804.

*Lösung:*

$$7\,777 = 7 \cdot 11 \cdot 101, \quad 10\,121\,804 = 2^2 \cdot 7 \cdot 11 \cdot 59 \cdot 557.$$

3. Alle zusammengesetzten Zahlen bis 10 000 enthalten in ihrer Primzahlfaktorisation eine Primzahl, die kleiner als 100 ist. Ist diese Aussage richtig?

*Lösung:*

Die Aussage folgt aus Satz 5.8. Ist eine natürliche Zahl kein Vielfaches einer Primzahl  $p$  mit  $p^2 \leq n$ , dann ist  $n$  eine Primzahl. 100 ist die Wurzel aus 10 000.

4. Führen Sie das Sieb des Erathostenes auf Papier und mit Hilfe des Computers für die Zahlen bis 180 durch!

*Lösung:*

Die Primzahlen kleiner als 180 sind gegeben durch

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, \\ 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179.$$

Sie müssen alle Vielfache bis einschließlich 13 untersuchen, da  $13^2 = 169$  und  $17^2 = 289$ .

Hier ein C++-Source-Code, der eine Liste der Primzahlen bis zu einer vorgegebenen Zahl ausgibt:

```

void erathostenes(unsigned long N, unsigned long &phi, bool *prim);

void main(void)
{
    unsigned long n, phi;

    cout << "Geben Sie N an:" << endl;
    cin >> n;

    bool *prim = new bool[n];

    erathostenes(n, phi, prim);

    // Ausgeben der Primzahlen
    cout << endl;
    for(int i=1; i < n; i++) {
        if(prim[i]) cout << i+1 << " ";
    }
    cout << endl;
    cout << " phi(" << n << ") = " << phi << endl;
}

void erathostenes(unsigned long N, unsigned long &phi, bool *prim)
{
    unsigned long wurzel = (unsigned long)sqrt(N);
    int i, j;

    // Initialisieren des Feldes
    for(i=0; i < N; i++) {
        prim[i] = true;
    }
    // Feststellen der Nichtprimzahlen
    // Auf Grund der Aussage, die die Grundlage des Siebs bildet,
    // muss nur bis wurzel(N) überprüft werden.
    //
    // Die Vielfachen werden gestrichen, übrig bleiben die
    // Primzahlen.
    // Auf count steht nach Durchlauf des Siebs die Anzahl der
    // Primzahlen kleiner oder gleich size; 1 ist keine Primzahl,
    // also wird sofort 1 subtrahiert.
    phi = N-1;
    for(i=2; i <= wurzel; i++) {
        for(j=2*i; j <= N; j+=i) {
            if (prim[j-1]) {
                prim[j-1] = false;
                phi--;
            }
        }
    }
}

```

Und hier eine Java-Klasse mit der gleichen Funktionalität:

```

public class Erathostenes {
    private static long phi;
    private static boolean[] prim;

    // Das Sieb als Konstruktor
    public Erathostenes (long N)

```

```

{
    int i, j;
    prim = new boolean[(int)N];
    long wurzel;

    // Initialisieren des Feldes
    for (i=0; i<N; i++) {
        prim[i] = true;
    }

    wurzel = (long)Math.sqrt(N);
    phi = N-1;
    for (i=2; i <= wurzel; i++) {
        for (j=2*i; j <= N; j+=i) {
            if (prim[j-1]) {
                prim[j-1] = false;
                phi--;
            }
        }
    }
} // Konstruktor

public static void main (String[] args) throws IOException {
    long N = readInt();
    Erathostenes sieb = new Erathostenes(N);
    System.out.println();
    for (int i=1; i<N; i++) {
        System.out.print(i+1+" ");
    }
    System.out.println();

    System.out.println("phi(" + N + ") = " + phi);
} // main

public static int readInt() throws IOException {
    int n;
    BufferedReader in = new BufferedReader(
        new InputStreamReader(System.in));

    System.out.print("Geben sie N an: ");
    n = Integer.parseInt(in.readLine());

    return n;
} // readInt
}

```

5. Berechnen Sie  $\text{ggT}(144, 89)$  und  $\text{ggT}(168, 9)$  mit dem Euklidischen Algorithmus und mit der rekursiven Methode!

*Lösung:*

144 und 89 sind relativ prim:

$$\begin{aligned}
 144 &= 1 \cdot 89 + 55, \\
 89 &= 1 \cdot 55 + 34, \\
 55 &= 1 \cdot 34 + 21, \\
 34 &= 1 \cdot 21 + 13, \\
 21 &= 1 \cdot 13 + 8, \\
 13 &= 1 \cdot 8 + 5, \\
 8 &= 1 \cdot 5 + 3, \\
 5 &= 1 \cdot 3 + 2, \\
 3 &= 1 \cdot 2 + 1, \\
 2 &= 2 \cdot 1.
 \end{aligned}$$

Rekursiv:

$$\text{ggT}(144, 89) = \text{ggT}(89, 55) = \text{ggT}(55, 43) = \dots = \text{ggT}(1, 0).$$

$\text{ggT}(168, 9) = 3$ :

$$\begin{aligned}
 168 &= 18 \cdot 9 + 6, \\
 9 &= 1 \cdot 6 + 3, \\
 6 &= 2 \cdot 3.
 \end{aligned}$$

Rekursiv:

$$\text{ggT}(168, 9) = \text{ggT}(9, 6) = \text{ggT}(6, 3) = \text{ggT}(3, 3) = \text{ggT}(3, 0) = 3.$$

6. Implementieren Sie den Euklidischen Algorithmus und testen Sie die Funktion mit den Ergebnissen aus Aufgabe 5!

*Lösung:*

```

int gcd(int a, int b)
{
    int h;

    do {
        h = a - (a/b)*b;
        a = b;
        b = h;
    } while (h !=0 );
    return a;
}

```

7. Lösen Sie die Gleichungen  $13x + 19y = 1\,000$  und  $6\,930x + 1\,098y = 18$ .

*Lösung:*

Die Lösung der ersten Gleichung ist  $x = 3\,000$ ,  $y = -2\,000$ . Die beiden Zahlen 13 und 19 sind relativ prim; wir lösen die Gleichung  $x \cdot 13 + y \cdot 19 = 1$  und multiplizieren anschließend die gefundenen Lösungen mit dem Faktor 1 000.

Die Rekursion verläuft wie folgt:

$$\begin{aligned}
 W_0 &= (19, 1, 0), W_1 = (13, 0, 1), \\
 q_0 &= 1, W_2 = W_0 - W_1 = (6, 1, -1), \\
 q_1 &= 6, W_3 = W_1 - 2 \cdot W_2 = (1, -2, 3).
 \end{aligned}$$

Damit ist  $(-2) \cdot 19 + 3 \cdot 13 = 1$  und die Lösung der ursprünglichen Aufgabe lautet  $x = 3\,000$ ,  $y = -2\,000$ .

Die zweite Gleichung ist lösbar, denn es ist  $\text{ggT}(6\,930, 1\,098) = 18$ . Die Rekursion verläuft wie folgt:

$$\begin{aligned} W_0 &= (6\,930, 1, 0), W_1 = (1\,098, 0, 1), \\ q_0 &= 6, W_2 = (342, 1, -6), \\ q_1 &= 3, W_3 = (72, -3, 19), \\ q_2 &= 4, W_4 = (54, 13, -82), \\ q_3 &= 1, W_5 = (18, -16, 101). \end{aligned}$$

Also ist die Lösung  $x = -16$ ,  $y = 101$ .

8. Implementieren Sie den verallgemeinerten Euklidischen Algorithmus und testen Sie die Funktion mit den Ergebnissen aus Aufgabe 7!

*Lösung:*

```
void extendedGcd(int a, int b, int ergebnis[3])
{
    int q, a1, a2, a3, b1, b2, b3, h1, h2, h3;
    a1 = a; a2 = 1; a3 = 0;
    b1 = b; b2 = 0; b3 = 1;

    do {
        q = a1/b1;
        h1 = a1 - q*b1;
        h2 = a2 - q*b2;
        h3 = a3 - q*b3;

        a1 = b1;
        a2 = b2;
        a3 = b3;
        b1 = h1;
        b2 = h2;
        b3 = h3;
    } while (h1 != 0);
    ergebnis[0] = a1;
    ergebnis[1] = a2;
    ergebnis[2] = a3;
}
```

9. Sind die Zahlen 45, 49 und 50 als  $25x + 35y$  darstellbar?

*Lösung:*

Es ist  $\text{ggT}(25, 35) = 5$ , also ist die Gleichung für die rechten Seiten 45 und 50 lösbar, mit den Lösungen  $x = 27$ ,  $y = -18$  und  $x = 30$ ,  $y = -20$ .

10. Stellen Sie die Verknüpfungstabellen für  $\odot$  und  $\oplus$  für  $m = 3$  und  $m = 6$  auf!

*Lösung:*

Die Verknüpfungstabellen für  $m = 3$  finden Sie in Tabelle 6.1, für  $m = 6$  in Tabelle 6.2.

11. Beweisen Sie die 3er und die 9er Regel: Eine Zahl  $n$  ist genau dann durch 3 bzw. 9 teilbar, wenn ihre Quersumme durch 3 bzw. 9 teilbar ist!

$\oplus$	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

$\odot$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

Tabelle 6.1: Die Verknüpfungstabellen für Aufgabe 10 im Fall  $m = 3$ 

$\oplus$	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	0
2	2	3	4	5	0	1
3	3	4	5	0	1	2
4	4	5	0	1	2	3
5	5	0	1	2	3	4

$\odot$	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

Tabelle 6.2: Die Verknüpfungstabellen für Aufgabe 10 im Fall  $m = 6$ 

*Lösung:*

Die Beweise verlaufen analog zur 11-er Regel. Es ist  $10^i \equiv 1 \pmod m$ , sowohl für  $m = 3$  als auch für  $m = 9$ . Dann ist für  $n = \sum_{i=0}^l a_i 10^i$

$$n \equiv \left( \sum_{i=0}^l a_i \right) \pmod m.$$

Wenn die Quersumme durch  $m$  teilbar ist, dann ist

$$\sum_{i=0}^l a_i \equiv 0 \pmod m.$$

12. Stellen Sie eine Wertetabelle für die Funktionen  $n^2 \pmod{11}$  und  $n^3 \pmod{11}$  auf und erstellen Sie den Graph!

*Lösung:*

Die Wertetabelle finden Sie in Tabelle 6.3; in 6.1 die Graphen.

Tabelle 6.3: Die Wertetabelle für die Funktionen  $n^2 \pmod{11}$  und  $x^3 \pmod{13}$  in Aufgabe 12

$n$	0	1	2	3	4	5	6	7	8	9	10
$n^2 \pmod{11}$	0	1	4	9	5	3	3	5	9	4	1
$n^3 \pmod{11}$	0	1	8	5	9	4	7	2	6	3	10

13. Verwenden Sie die Lösung der Aufgabe 12, um die zweite modulare Wurzel aus  $n = 4$  und  $n = 5$  sowie die dritte modulare Wurzel aus  $n = 2$  zu bestimmen.

*Lösung:*

Die zweite Wurzel aus 4 sind  $n = 2$  und  $n = 9$ .

Die zweite Wurzel aus 5 sind  $n = 4$  und  $n = 7$ .

Die dritte Wurzel aus 2 ist  $n = 7$ .

14. Welche Zahlen haben zum Modul 8 eine modulare Inverse? Geben Sie die Inversen an!

*Lösung:*

Außer der 1 die Zahlen zwischen 2 und 7, die relativ prim mit 8 sind; das sind dann 3, 5 und 7.



Abbildung 6.1: Der Graph der Funktionen  $n^2 \bmod 11$  (links) und  $n^3 \bmod 11$  (rechts) für Aufgabe 12

Mit dem verallgemeinerten Euklidischen Algorithmus erhält man  $-8 + 3 \cdot 3 = 1$ ; also ist die modulare Inverse von 3 zum Modul 8 gleich 3.

Für 5 ergibt sich  $2 \cdot 8 + (-3) \cdot 5 = 1$ ; also ist die modulare Inverse von 5 zum Modul 8 gegeben durch  $-3$  bzw. 5.

Für 7 ergibt sich  $8 + (-1) \cdot 7 = 1$ , also ist die modulare Inverse von 7 zum Modul 8 gegeben durch  $-1$  bzw. 7.

15. Beweisen Sie, dass für die ganzen Zahlen  $a, b, q, r$  aus  $a = q \cdot b + r$  die Gleichung  $\text{ggT}(a, b) = \text{ggT}(b, r)$  folgt.

*Lösung:*

Es reicht zu zeigen, dass die Menge der Teiler von  $a$  und  $b$  mit der von  $b$  und  $r$  übereinstimmt; dann muss auch  $\text{ggT}(a, b) = \text{ggT}(b, r)$  gelten.

Angenommen,  $d$  ist eine Zahl mit  $d|a$  und  $d|b$ . Dann teilt  $d$  auch  $r = a - b \cdot q$ .

Angenommen,  $c$  ist eine Zahl mit  $c|b$  und  $c|r$ . Dann teilt  $c$  auch  $b \cdot q + r = a$ .

16. Berechnen Sie  $28^{12} \bmod 13$ ,  $3^{15} \bmod 13$ ,  $15^{83} \bmod 13$ ,  $17^{20} \bmod 50$  und  $49^{28} \bmod 58$ .

*Lösung:*

$28^{12} \bmod 13 \equiv 1$ , denn 28 und 13 sind relativ prim, der kleine Fermat'sche Satz kann angewandt werden.

Es gilt  $3^{12} \equiv 1 \bmod 13$  und damit  $3^{15} \equiv 3^3 \bmod 13 \equiv 1 \bmod 13$ .

$15^{83} \equiv 7 \bmod 13$ , denn mit dem kleinen Satz von Fermat gilt

$$15^{83} = 15^{72} \cdot 15^{11} \equiv (15^{12})^6 \cdot 15^{11} \bmod 13 \equiv 15^{11} \bmod 13.$$

$15^{12} \equiv 1 \bmod 13$ ; also ist die Lösung gegeben durch die modulare Inverse von 15 zum Modul 13, die durch 7 gegeben ist. Multiplikation mit 7 entspricht der Division durch 15, es ist  $\text{ggT}(13, 15) = 1$ .

$17^{20} \bmod 50 \equiv 1$ , denn 17 und 50 sind relativ prim, und  $\varphi(50) = 20$ ; der Satz von Euler kann angewandt werden.

Auch hier kann der Satz von Euler angewandt werden:  $49^{28} \bmod 58 \equiv 1$ , denn  $\varphi(58) = 28$ .

17. Berechnen Sie die Werte der Euler'schen Funktion für  $41 \leq n \leq 60$ .

*Lösung:*

$$\begin{aligned}
\varphi(41) &= 40, \varphi(42) = \varphi(2 \cdot 3 \cdot 7) = 12, \varphi(43) = 42, \\
\varphi(44) &= \varphi(2^2 \cdot 11) = 20, \varphi(45) = \varphi(3^2 \cdot 5) = 30, \varphi(46) = \varphi(2 \cdot 23) = 22, \\
\varphi(47) &= 46, \varphi(48) = \varphi(2^4 \cdot 3) = 16, \varphi(49) = \varphi(7^2) = 42, \\
\varphi(50) &= \varphi(2 \cdot 5^2) = 20, \varphi(51) = \varphi(3 \cdot 17) = 32, \varphi(52) = \varphi(2^2 \cdot 13) = 24, \\
\varphi(53) &= 52, \varphi(54) = \varphi(2 \cdot 3^3) = 18, \varphi(55) = \varphi(5 \cdot 11) = 40, \\
\varphi(56) &= \varphi(2^3 \cdot 7) = 24, \varphi(57) = \varphi(3 \cdot 19) = 36, \varphi(58) = \varphi(2 \cdot 29) = 28, \\
\varphi(59) &= 58, \varphi(60) = \varphi(2^2 \cdot 3 \cdot 5) = 16.
\end{aligned}$$

18. Implementieren Sie die Tausch-Chiffre und führen Sie sie mit den Schlüsseln (5, 21) und (15, 7) durch für die Nachricht „mathematik fuer informatiker“!

*Lösung:*

Für den Schlüssel (5, 21) wird „mathematik“ wie folgt abgebildet. Zuerst auf die Zahlen

$$12, 0, 19, 7, 4, 12, 0, 19, 8, 10;$$

dann die Tauschchiffre

$$19, 1, 10, 18, 7, 19, 1, 10, 13, 3,$$

und als Zeichenkette „tbkshtbknds“.

Das Wort „fuer“ wird auf die Zahlen 5, 20, 4, 17 abgebildet; nach der Tauschchiffre ist 2, 5, 7, 20 und als Zeichenkette „cfhu“.

Für „informatik“ ergeben sich die Zahlen

$$8, 13, 5, 14, 17, 12, 0, 19, 8, 10$$

und nach der Tauschchiffre

$$13, 14, 2, 9, 20, 19, 1, 10, 13, 3.$$

Das entspricht der Zeichenkette „nocjtbknd“

Für die Schlüssel (15, 7) ergibt sich für „mathematik“ die Tauschchiffre

$$7, 1, 4, 24, 3, 7, 1, 4, 5, 19;$$

als Zeichenkette „hbeydhbef“.

Das Wort „fuer“ wird auf die Tauschchiffre 2, 5, 7, 20 und die Zeichenkette „cfhu“ abgebildet.

Für „informatik“ ergibt sich die Tauschchiffre

$$5, 14, 10, 21, 16, 7, 1, 4, 5, 19$$

und als Zeichenkette „fokvqhbf“.

Und hier der Quellcode:

```

// Funktion, die aus char zwischen a und z eine zahl zwischen 0 und 25 macht
int c2i(const char c)
{
    return int(c)-97;
}

// Funktion, die aus int zwischen 1 und 26 char zwischen a und z macht
char i2c(int i)
{

```

```

    return char(i+97);
}

// Funktion, die den Eingabe-String in ein Integer-Feld umwandelt
void s2i(int laenge, const char satz[], int isatz[])
{
    for (int i = 0; i<laenge; i++)
        isatz[i] = c2i(satz[i]);
}

// Funktion, die ein Integer-Feld in einen String umwandelt
void i2s(int laenge, const int chiffre[], char satz[])
{
    for (int i = 0; i<laenge; i++)
        satz[i] = i2c(chiffre[i]);
}

// Funktion, die die Verschiebe-Chiffre mit S durchfuehrt
// alles modulo 26.
void verschiebeChiffre(int laenge, int s, int satz[])
{
    for (int i=0; i<laenge; i++)
    {
        satz[i] = (satz[i] + s) % 26;
        // Beim Dechiffrieren wird die negative Verschiebung in diese
        // Funktion eingesetzt. Dadurch können negative Restklassenvertreter
        // auftauchen. Abfangen und 26 dazu addieren.
        if (satz[i] < 0 ) satz[i] += 26;
    }
}

// Funktion, die multipliziert, modulo 26.
void mult(int laenge, int t, int satz[])
{
    for (int i=0; i<laenge; i++)
        satz[i] = (satz[i]*t) % 26;
}

// Tausch-Chiffre: zuerst verschieben, dann multiplizieren.
void tauschChiffre(int laenge, int s, int t, int chiffre[])
{
    verschiebeChiffre(laenge, s, chiffre);
    mult(laenge, t, chiffre);
}

// Funktion, die eine Tauschchiffre dechiffriert.
// s ist die Verschiebung, t der multiplikative Faktor.
// Als Faktor muss die multiplikative Inverse zum Modul 26 verwendet werden.
// Chiffrieren und deChiffrieren unterscheiden sich durch die Reihenfolge.
// Bei der Dechiffrierung wird zuerst multipliziert, dann -s abgezogen!
void tauschDeChiffre(int laenge, int s, int t, int chiffre[])
{
    mult(laenge, t, chiffre);
    verschiebeChiffre(laenge, s, chiffre);
}

```

19. Wie können Sie eine Tausch-Chiffre mit Schlüssel  $(s, t)$  angreifen, wenn Sie bereits für zwei Buchstabenpaare die Zuordnung kennen? Führen Sie das gefundene Verfahren durch für das

Paar  $c \mapsto d, h \mapsto s$ ! Dechiffrieren Sie mit dem gefundenen Schlüssel den Geheimtext „zds-wvmc“!

*Lösung:*

Durch die zwei bekannten Paare kann ein modulares lineares Gleichungssystem aufgestellt werden. Für die Funktion  $f(n)$  sind die beiden Werte  $f(2) = 3$  und  $f(7) = 18$  bekannt. Dann sind  $s$  und  $t$  gesucht mit

$$\begin{aligned} 2 \cdot t + s &\equiv 3 \pmod{26}, \\ 7 \cdot t + s &\equiv 18 \pmod{26}. \end{aligned}$$

Auflösen der ersten Gleichung nach  $s$  ergibt

$$s \equiv (3 - 2 \cdot t) \pmod{26}.$$

Das können Sie in die zweite Gleichung einsetzen:

$$7 \cdot t + 3 - 2 \cdot t \equiv 18 \pmod{26}.$$

Dann erfüllt  $t$  die Gleichung

$$5 \cdot t \equiv 15 \pmod{26}.$$

Die multiplikative Inverse zu 5 ist die Zahl 21. Wir multiplizieren beide Seiten der Gleichung damit und erhalten

$$\begin{aligned} t &\equiv (21 \cdot 15) \pmod{26} \\ &\equiv 315 \pmod{26} \\ &\equiv 3. \end{aligned}$$

Dann ist  $s \equiv (3 - 6) \pmod{26} \equiv (-3) \pmod{26} \equiv 23$ .

Es wurde der Schlüssel  $(23, 3)$  verwendet.

20. Für das Teilmengen-Summenproblem gibt es einen Empfänger mit privatem Schlüssel 1, 2, 5, 11, 32, 87, 141,  $w = 901$ ,  $m = 1234$  und öffentlichem Schlüssel 901, 568, 803, 39, 450, 645, 1173. Ver- und entschlüsseln Sie die Nachricht „mathematik fuer informatiker“!

*Lösung:*

Die Teilmengen-Summen Chiffre für das Wort „mathematik“:

$$3131, 2642, 2722, 1508, 3092, 3131, 2642, 2722, 2681, 3326.$$

Die Teilmengen-Summen Chiffre für das Wort „fuer“:

$$2564, 3895, 3092, 2917.$$

Die Teilmengen-Summen Chiffre für das Wort „informatiker“:

$$2681, 2603, 2564, 3776, 2917, 3131, 2642, 2722, 2681, 3326, 3092, 2917.$$

Hier der Quelltext der dafür benutzten C++-Funktionen. Für die 7-Bitdarstellung der Buchstaben wird die STL-Klasse `bitset` verwendet.

```
unsigned long knapsack(char wort, unsigned long publicKey[7])
{
    unsigned long ch;
    int i;
    bitset<7> bits(wort);
```

```
    ch = 0;
    for (i=0; i<7; i++)
        ch += key[i]*bits[i];
    return ch;
}

// Es wird davon ausgegangen, dass die im privaten Schlüssel
// abgelegten Zahlen // bereits so angeordnet sind,
// dass jede größer ist als die Summe aller ihrer Vorgänger.
unsigned long deKnapsack(unsigned long Bprime, unsigned long privateKey[7])
{
    int i, first;
    unsigned long summe;
    bitset<7> bits; // Alle bits sind auf Null gesetzt!

    // Zuerst den kleinsten privaten Schlüssel finden
    // in der Summe vorkommt.
    first = 6;
    summe = privateKey[first];
    while (Bprime <= summe) {
        summe = privateKey[first-1];
        first--;
    }
    bits.set(6-first);
    if (Bprime > summe )
        for (i=first-1; i>=0; i--) {
            if (Bprime >= summe+privateKey[i]) {
                summe += privateKey[i];
                bits.set(6-i);
            }
        }
    // Jetzt aus der Bitfolge wieder eine natürliche Zahl machen
    return bits.to_ulong();
}

void summenChiffre(int laenge, unsigned long publicKey[7],
                  char satz[], int chiffre[])
{
    int i;
    for (i=0; i<laenge; i++)
        chiffre[i] = knapsack(satz[i], key);
}

void summenDeChiffre(int laenge, unsigned long winv, unsigned long m,
                    unsigned long privateKey[7], unsigned long chiffre[],
                    char satz[])
{
    int i;
    unsigned long asciiCode, Bprime;

    for (i=0; i<laenge; i++) {
        Bprime = (chiffre[i]*winv) % m;
        asciiCode = deKnapsack(Bprime, privateKey);
        satz[i] = char(asciiCode);
    }
}
```

21. Bilden Sie ein eigenes Kryptosystem auf der Basis des Teilmengen-Summenproblems und testen Sie dieses System mit den Nachrichten „hanser“ und „mathematik fuer informatiker“!

*Lösung:*

Für ein Kryptosystem nach dem Teilmengen-Summenproblem benötigen Sie als aller erstes zwei natürliche Zahlen  $m$  und  $w$ , die relativ prim sind. Dann gibt es für  $w$  zum Modul  $m$  die modulare Inverse  $w^{-1}$ .

Anschließend müssen Sie für jeden Teilnehmer 7 Zahlen  $p_0, p_1, \dots, p_6$  suchen, die

$$\sum_{i=0}^k p_i < p_{k+1}, k = 0, \dots, 5.$$

erfüllen. Diese bilden den privaten Schlüssel für den Teilnehmer. Als öffentlichen Schlüssel veröffentlichen Sie anschließend die 7 Zahlen  $a_0, a_1, \dots, a_6$  mit

$$a_k = (w \cdot p_i) \bmod m.$$

Als Modul wählen wir jetzt  $m = 1\,026$  und als  $w = 511$ . Überzeugen Sie sich mit Hilfe des verallgemeinerten Euklidischen Algorithmus, es ist  $\text{ggT}(1026, 511) = 1$ ; als modulare Inverse erhalten Sie  $w^{-1} = 769$ .

Als privaten Schlüssel eines neuen Teilnehmers werden die Zahlen

$$p_0 = 3, p_1 = 4, p_2 = 11, p_3 = 23, p_4 = 45, p_5 = 87, p_6 = 256$$

gewählt. Dann ist der öffentlichen Schlüssel für diesen Teilnehmer

$$a_0 = 507, a_1 = 1018, a_2 = 491, a_3 = 467, a_4 = 423, a_5 = 339, a_6 = 514.$$

Für die Nachricht „hanser“ ergibt sich dann die Chiffre

$$1992, 2039, 2754, 2869, 2462, 2355.$$

Für „mathematik“ ist die Chiffre

$$2929, 2039, 2439, 1992, 2462, 2929, 2039, 2439, 2506, 2845;$$

„fuer“ wird zu 2287, 2953, 2462, 2355 und „informatiker“ zu

$$2506, 2754, 2287, 3268, 2355, 2929, 2039, 2439, 2506, 2845, 2462, 2355.$$

22. Übertragen Sie für den öffentlichen Schlüssel  $n = 34\,571 = 181 \cdot 191$ ,  $e = 7\,901$  und für den privaten Schlüssel  $d = 11\,501$  die Nachricht „mathematik fuer informatiker“ mit einem RSA-System!

*Lösung:*

Ich hoffe, Sie haben nicht versucht, diese Aufgabe per Hand zu lösen. Ein Problem bei der Implementierung sind die großen Zahlen, die auftreten. Eine Lösung wäre, die Potenzen so zu berechnen, dass nach jeder Multiplikation sofort wieder die Restklasse bestimmt wird.

In Java können Sie die Klasse `BigInteger` für beliebig große Zahlen verwenden, die eine Funktion `modPow` zur Verfügung stellt.

In C++ wäre die Potenz dann wie folgt implementierbar:

```
// Produkt als sukzessives Addieren mit zwischenzweitleichem Rest.
unsigned long produktMod(unsigned long a, unsigned long b, unsigned long m)
{
```

```

unsigned long i, summe;

summe = 0;
for (i=0; i<b; i++)
    summe = (summe + a) % m;

return summe;
}
// Potenz als sukzessives Multiplizieren mit zwischenzeitlichem Rest.
unsigned long potenzMod(unsigned long a, unsigned long b, unsigned long m)
{
    unsigned long i, potenz;

    potenz = 1;
    for (i=0; i<b; i++)
        potenz = produktMod(potenz, a, m);

    return potenz;
}

```

Das RSA-System ist dann etwa so implementierbar:

```

void rsaChiffre(int laenge, unsigned long publicKey, unsigned long m,
               unsigned long chiffre[])
{
    int i;

    for (i=0; i<laenge; i++) {
        chiffre[i] = potenzMod(chiffre[i], publicKey, m);
    }
}

void rsaDeChiffre(int laenge, unsigned long privateKey, unsigned long m,
                  unsigned long chiffre[])
{
    int i;

    for (i=0; i<laenge; i++) {
        chiffre[i] = potenzMod(chiffre[i], privateKey, m);
    }
}

```

Die RSA-Chiffre für das Wort „mathematik“:

3 165, 22 783, 31 817, 24 968, 16 790, 3 165, 22 783, 31 817, 21 486, 23 456.

Die RSA-Chiffre für das Wort „fuer“:

13 390, 32 585, 16 790, 27 537.

Die RSA-Chiffre für das Wort „informatiker“:

21 486, 28 638, 13 390, 5 441, 27 537, 3 165, 22 783, 31 817, 21 486, 23 456, 16 790, 27537.

23. Bilden Sie ein eigenes RSA-Kryptosystem und testen Sie dieses System mit den Nachrichten „hanser“ und „mathematik fuer informatiker“!

*Lösung:*

Zuerst müssen Sie zwei Primzahlen auswählen, beispielsweise  $p = 167$  und  $q = 97$ . Dann ist  $m = p \cdot q = 16199$  und  $\varphi(m) = (p-1)(q-1) = 15936$ .

Für die zu  $\varphi(m)$  relativ prime Zahl kann beispielsweise  $d = 41$  gewählt werden. Mit dem verallgemeinerten Euklidischen Algorithmus kann dazu die modulare Inverse zum Modul  $\varphi(m)$  berechnet werden; es ist  $e = 7385$ .

Die RSA-Chiffre für das Wort „hanser“ lautet dann:

12 722, 6 305, 9 007, 1 049, 12 847, 11 150.

Die RSA-Chiffre für das Wort „mathematik“:

15 023, 6 305, 127, 12 722, 12 847, 15 023, 6 305, 127, 1 447, 3 261.

Die RSA-Chiffre für das Wort „fuer“:

11 366, 12 759, 12 847, 11 150.

Die RSA-Chiffre für das Wort „informatiker“:

1 447, 9 007, 11 366, 14 854, 11 150, 15 023, 6 305, 127, 1 447, 3 261, 12 847, 11 150.

24. Übertragen Sie für  $p = 7\,919$ ,  $a = 511$ , dem öffentlichen Schlüssel  $y = 4\,143$  und privatem Schlüssel  $x = 73$  eines ElGamal-Systems die Nachricht „mathematik fuer informatiker“!

*Lösung:*

Ich hoffe, Sie haben genauso wie bei den Aufgaben zu RSA nicht versucht, diese Aufgabe per Hand zu lösen. Ein Problem bei der Implementierung sind die großen Zahlen, die auftreten. Eine Lösung wäre, die Potenzen so zu berechnen, dass nach jeder Multiplikation sofort wieder die Restklasse bestimmt wird. Die Lösung besteht wie beim RSA-Verfahren entweder in der Verwendung der Klasse `BigInteger` für Java oder die C++-Funktionen, die schon in Aufgabe 22 vorgestellt wurden.

Das ElGamal-Verfahren ist dann etwa so implementierbar:

```
void elGamalChiffre(int laenge, unsigned long p, unsigned long a,
                  unsigned long y, unsigned long r,
                  unsigned long chiffre[], unsigned long &z)
{
    // Die Nachricht besteht aus der chiffre und der Zahl z!
    int i;
    unsigned long yr;

    yr = potenzMod(y,r,p);

    for (i=0; i<laenge; i++)
        chiffre[i] = produktMod(chiffre[i],yr,p);

    z = potenzMod(a,r,p);
}

void elGamalDeChiffre(int laenge, unsigned long privateKey, unsigned long p,
                    unsigned long chiffre[], unsigned long z)
{
    int i;
    long potenz, inverse;

    potenz = potenzMod(z,privateKey,p);
    inverse = modulareInverse(potenz, p);
```

```

for (i = 0; i < laenge; i++)
    chiffre[i] = produktMod(chiffre[i], inverse, p);
}

```

Die ElGamal-Chiffre für das Wort „mathematik“:

791, 5 063, 6 218, 2 571, 3 639, 791, 5 063, 6 218, 2 215, 1 503.

Die ElGamal-Chiffre für das Wort „fuer“:

3 283, 5 862, 3 639, 6 930.

Die ElGamal-Chiffre für das Wort „informatiker“:

2 215, 435, 3 283, 79, 6 930, 791, 5 063, 6 218, 2 215, 1 503, 3 639, 6 930.

25. Bilden Sie ein eigenes ElGamal-Kryptosystem und testen Sie dieses System mit den Nachrichten „hanser“ und „mathematik fuer informatiker“!

*Lösung:*

Sie benötigen eine Primzahl  $p$  und eine dazu relativ prime Basis  $a$ . Möglich wäre  $p = 6\,133$  und  $a = 471$ .

Für den Empfänger soll der private Schlüssel  $x = 99$  gewählt sein. Daraus ergibt sich mit  $y \equiv a^x \pmod{p}$  der öffentliche Schlüssel  $y = 1\,071$ . Zusätzlich wird noch die zufällige Zahl  $r = 7$  ausgewählt.

Die zusätzlich übertragene Zahl ist  $z = 5\,762$ .

Die ElGamal-Chiffre für das Wort „hanser“ lautet:

2 355, 3 199, 4 260, 2 781, 4 469, 5 530.

Die ElGamal-Chiffre für das Wort „mathematik“:

876, 3 199, 32, 2 355, 4 469, 876, 3 199, 32, 5 739, 241.

Die ElGamal-Chiffre für das Wort „fuer“:

1 720, 3 416, 4 469, 5 530.

Die ElGamal-Chiffre für das Wort „informatiker“:

5 739, 4 260, 1 720, 1 511, 5 530, 876, 3 199, 32, 5 739, 241, 4 469, 5 530.