

# Kryptographie I - Zusammenfassung

Autoren: Linda Schneider und Julian Kotzur

## 1 Einfache Substitutionschiffren

### CAESAR-Verschlüsselung:

1. Verschlüsselung: Alle Zeichen werden um  $k$  Plätze nach **links** verschoben.
2. Entschlüsselung: Alle Zeichen werden um  $k$  Plätze nach rechts verschoben.
3. Angriff: Häufigkeitsanalyse oder alle Möglichkeiten durchprobieren.
4. Beispiel Caesar 3:

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C |

### MASC-Verschlüsselung

1. Verschlüsselung: Oben Alphabet, unten am Anfang Schlüsselwort eintragen. **Doppelte Zeichen werden eliminiert**, am Ende wird ausgehen vom letzten Buchstaben das Alphabet zirkulär fortgesetzt.
2. Entschlüsselung: Bijektive Umkehrfunktion.
3. Angriff: Häufigkeitsanalyse bzw. Textteile erraten und abhängig vom zirkulären Alphabet auffüllen.
4. Beispiel: Key = Erlangen, Text: Baum → Reyp

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z |
| E | R | L | A | N | G | H | I | J | K | M | O | P | Q | S | T | U | V | W | X | Y | Z | B | C | D | F |

### TRANSMAT-Verschlüsselung

1. Verschlüsselung:  $m \times n$  **Matrix mit  $m$  Zeilen und  $n$  Spalten**, in die der Text eingetragen wird. **Es wird spaltenweise ausgelesen**.
2. Schlüssel besteht aus zwei natürlichen Zahlen  $m$  und  $n$ .
3. Entschlüsselung: Chiffretext spaltenweise in  $m \times n$  - Matrix schreiben und **zeilenweise** auslesen.
4. Angriff: verschiedene Matrizen ausprobieren

### TRANSSPA-Verschlüsselung

1. Schlüssel: Wort mit  $n$  Buchstaben. Tabelle anlegen, wobei pro Buchstaben eine Spalte existiert
2. Verschlüsselung: Text **zeilenweise** in die Tabelle eintragen und anschließend **spaltenweise nach Reihenfolge der Buchstaben des Schlüssels** ausgelesen. Bei gleichen Buchstaben geschieht dies von links nach rechts.
3. Entschlüsselung: Chiffretext spaltenweise in eine Tabelle mit  $n$  Spalten eintragen und mit richtiger Permutation zeilenweise auslesen.
4. Angriff: Auffüllzeichen betrachten oder Schlüssellänge raten

### ALBC-1-Verschlüsselung

1. key:  $a, b \in \mathbb{N}$  mit  $\text{ggT}(a; N) = 1$  ( $N = \text{Alphabetlänge}$ ).
2. Verschlüsselungsfunktion:  $f(x) = ax + b \pmod N$
3. Entschlüsselungsfunktion:  $f^{-1}(x) = cx + d \pmod N$
4. Angriff: **Erraten von 2 Buchstaben. Damit Gleichungssystem aufstellen und  $c, d$  berechnen.**

### ALBC-k-Verschlüsselung

1. key:  $x_1, \dots, x_{k^2+k}$  mit  $\text{ggT}(\det A, N) = 1$  ( $N = \text{Alphabetlänge}$ ,  $A = k \times k$  -Matrix)
2. Verschlüsselungsfunktion:  $f(y_1, \dots, y_k) = Ay + (x_{k^2+1}, \dots, x_{k^2+k}) \pmod N$
3. Entschlüsselungsfunktion ist Umkehrfunktion
4. Angriff: **Erraten von  $k^2 + k$  Buchstaben**, damit entstandenes Gleichungssystem lösen.

### Operations-Modi

1. ECB-Modus: Text wird in **Blöcke gleicher Länge** aufgeteilt, die jeweils einzeln verschlüsselt werden.  
Nachteil: Regelmäßigkeiten im Klartext führen zu Regelmäßigkeiten im Chiffretext.
2. CBC-Modus: **Rekursive Verschlüsselung** eines Text anhand einer Formel.  $k$  nicht durch Häufigkeitsanalyse bestimmbar  
→ Beispiel Caesar: Formel:  $b_i = b_{i-1} + a_i + k \pmod{26}$

### STROM-Chiffrierung

1. Es werden nur Großbuchstaben berücksichtigt!
2. Der Schlüssel besteht aus einer **Folge von Großbuchstaben  $k_1, k_2, \dots$ , die mindestens so lang ist wie der verschlüsselte Text**.
3. Verschlüsselungsfunktion:  $f(a_i, k_i) = a_i + k_i \pmod{26}$
4. Entschlüsselungsfunktion:  $g(b_i, k_i) = b_i - k_i \pmod{26}$
5. Vernam / One-Time-Pad: Alphabet und Schlüssel binär, Addition ohne Übertrag.
6. AUTOKEY: Der key ist ein Schlüsselwort kleiner als der Text, an den der Klartext angehängt wird.
7. Beispiel:

|           |   |   |   |   |   |   |   |   |   |   |   |   |
|-----------|---|---|---|---|---|---|---|---|---|---|---|---|
| Text      | D | A | R | L | E | K | S | L | E | B | E | N |
| Schlüssel | E | R | L | A | N | G | H | I | J | K | M | O |
| Chiffre   | H | R | C | L | R | Q | Z | T | N | L | Q | B |

## 2 Klassische Chiffrierverfahren

### VIGENÈRE-Verschlüsselung

- Schlüsselwort  $k_1, k_2, \dots, k_n$ , welches **periodisch** auf den Text angewandt wird.
- Verschlüsselungsfunktion:  $b_i = a_i + k_i \text{ mod } 26$ .
- Entschlüsselungsfunktion:  $a_i = b_i - k_i \text{ mod } 26$ .
- Angriff: Kasiski-Test.
- "Periodisches Caesar" Beispiel:

|           |   |   |   |   |   |   |   |   |   |   |   |   |
|-----------|---|---|---|---|---|---|---|---|---|---|---|---|
| Text      | D | A | R | L | E | K | S | L | E | B | E | N |
| Schlüssel | B | A | U | M | B | A | U | M | B | A | U | M |
| Chiffre   | E | A | L | X | F | K | M | X | F | B | Y | Z |

### Kasiski-Test

- Suche im verschlüsselten Text Zeichenketten die mehrfach auftreten. Daraus kann folgen, dass gleicher Klartext gleich verschlüsselt wurde.
- Suche weitere **gleiche Zeichenketten** der Länge  $n$  (oder ähnliche) und vermerke die Abstände  $j_1 - i_1, j_2 - i_2, \dots$ . Stimmt die Erklärung aus (1), so erhalten wir

$$n | ggT(j_1 - i_1, j_2 - i_2, \dots)$$

Bei entsprechend vielen gleichen kurzen Zeichenketten kann man hoffen, dass

$$n = ggT(j_1 - i_1, j_2 - i_2, \dots)$$

gilt und man so die **Schlüssellänge  $n$  bestimmt** hat.

⇒ Zur weiteren Entschlüsselung teilt man den Chiffretext in eine Tabelle mit  $n$  Spalten auf, welche nun **CAESAR- $k_j$ -chiffriert** sind, und führt eine Häufigkeitsanalyse innerhalb der Spalten durch.

### PLAYFAIR-Verschlüsselung

- Alphabet mit **25 Großbuchstaben**, wobei J = I.
- Eine Code-Wort wird in eine  $5 \times 5$ -Matrix zeilenweise eingetragen. Doppelte Buchstaben werden gedarlegt! Die restliche Matrix wird alphabetisch mit den noch fehlenden Buchstaben ergänzt.
- Der Ausgangstext wird in **Bigramme** unterteilt, wobei der Buchstabe X eingefügt wird, falls ein Bigramm aus 2 gleichen Buchstaben besteht.
- Verschlüsselung: Bigramm in Matrix suchen und nach folgenden 3 Regeln umschreiben:
  - Gleiche Zeile: Eins nach rechts (zirkulär).**
  - Gleiche Spalte: Eins nach unten (zirkulär).**
  - Sonst: Quadrat bilden und Ecktausch zeilenweise nach Bigrammreihenfolge.**
- Für die Entschlüsselung Regeln invers anwenden.
- Beispiel:

|   |   |   |   |   |
|---|---|---|---|---|
| E | R | L | A | N |
| G | B | C | D | F |
| H | J | K | M | O |
| P | Q | S | T | U |
| V | W | X | Y | Z |

### ADFGVX-Chiffrierung

- Das Klartextalphabet besteht aus 36 Zeichen, den Buchstaben  $A, \dots, Z$  und den Ziffern  $0, 1, \dots, 9$ . Das Chiffretextalphabet aus den Zeichen  $A, D, F, G, V, X$ .
- Schlüssel: **Erstellen einer  $6 \times 6$  und Wahl eines Permutationsschlüssels der Länge  $n$ .**
- Verschlüsselung: Umwandlung des Klartextes in die Zeichenpaare des Chiffrealphabets. Anschließend wird der Chiffretext zeilenweise in  $n$  Spalten aufgeteilt und permutiert. Am Schluss wird der Chiffretext spaltenweise ausgelesen.
- Entschlüsselung: Mit **erstem Schlüssel Matrix** erstellen. Mit dem **zweiten Schlüssel Buchstabenreihenfolge** bestimmen. Durch Anzahl Zeichen mod Schlüssellänge **Füllbits** berechnen, mit Teilen und Aufrunden Zeilenanzahl berechnen. Text Nach Reihenfolge der Spalten eintragen und zeilenweise mit Matrix entschlüsseln.

## 3 Grundlagen

### Definition 3.1 (modulo-Definition)

Für  $a \in \mathbb{Z}$  und  $b \in \mathbb{N}$  wird 'a modulo b' definiert als

$$a \text{ mod } b = a - \lfloor \frac{a}{b} \rfloor b.$$

**Tipp: modulo im Taschenrechner:**

$$A / B = X, Y \rightarrow X, Y - X = 0, Y \rightarrow 0, Y \cdot B = \text{Rest}$$

### Naives Faktorisierungsverfahren

- **Ziel: kleinsten Teiler von  $n$  finden, bzw. Primzahlnachweis.**
- Betrachtet werden alle Primzahlen  $p_i : p_i \leq \sqrt{n}$ .
- Wenn ein  $p_i$  existiert mit:  $n \text{ mod } p_i = 0 \Rightarrow p_i$  kleinster Teiler.
- Wenn kein  $p_i$  existiert  $\Rightarrow n$  ist eine Primzahl.

### Größter gemeinsamer Teiler

#### Eigenschaften des ggT

Seien  $a, b, c \neq 0$  ganze Zahlen.

- $a|c, b|c$  und  $ggT(a, b) = 1 \Rightarrow ab|c$
- $a|bc$  und  $ggT(a, c) = 1 \Rightarrow a|b$
- Ist  $d = ggT(a, b)$ , schreibt man  $a = da', b = db'$ , so gilt  $ggT(a', b') = 1$
- Die Gleichung  $ax + by = c$  ist genau dann lösbar, wenn  $ggT(a, b)|c$  gilt.**

### Kleinstes gemeinsames Vielfaches

- $kgV(m, n)$ : Kleinstes gemeinsames Vielfaches von  $m$  und  $n$ .
- **Berechnung  $kgV(m, n)$  = Multiplikation höchster Potenzen jeglicher Primfaktoren.**

### Nützliche Tipps:

- $m, n \in \mathbb{Z}$  sind teilerfremd, wenn  $ggT(m, n) = 1$  gilt.
- **$ggT(m, n)$  = Multiplikation niedrigster Potenzen gemeinsamer Primfaktoren**
- Primfaktoren  $m+n$ : Ausklammern gemeinsamer Primfaktoren und Primfaktorzerlegung des Rests.

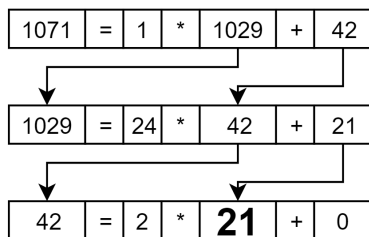
## Der euklidische Algorithmus

Seien  $a, b \in \mathbb{N}_0$  gegeben. Wir setzen  $a_0 = a$  und  $a_1 = b$ .

1. Ist  $a_{i+1} = 0$ , so ist man fertig.
2. Ist  $a_{i+1} > 0$ , so dividiert man  $a_i$  durch  $a_{i+1}$  und erhält den Quotienten  $q_i$  und den Rest  $a_{i+2}$ .

Dann gilt  $ggT(a, b) = a_n$ .

Beispiel:



## Der erweiterte euklidische Algorithmus

### Satz 3.2

Zu  $a, b \in \mathbb{Z}$  gibt es  $x, y \in \mathbb{Z}$  mit  $ggT(a, b) = xa + yb$ .

Seien  $a, b \in \mathbb{N}_0$  gegeben. Man führe den euklidischen Algorithmus aus und berechne in jedem Schritt zusätzlich rekursiv  $x_i, y_i$  mit den folgenden Formeln:

$$x_i = x_{i-2} - q_i \cdot x_{i-1} \quad \text{und} \quad y_i = y_{i-2} - q_i \cdot y_{i-1}$$

Hierbei werden  $x, y$  wie folgt initialisiert:

$$x_0 = 1, \quad x_1 = 0, \quad y_0 = 0, \quad y_1 = 1.$$

→ **Test:**  $a \cdot y \equiv 1 \pmod b$  (**y ist Inverses!**)

Beispiel:

| Zeile   | a    | q  | x    | y          | Berechnung                    |
|---|------|----|------|------------|-------------------------------|
| 0   | 1077 | -  | 1    | 0          | Vorinitialisiert              |
| 1   | 1029 | -  | 0    | 1          | Vorinitialisiert              |
| 2   | 48   | 1  | 1    | -1         | $x_2 = x_0 - (q_2 \cdot x_1)$ |
| 3   | 21   | 21 | -21  | 22         | $x_3 = x_1 - (q_3 \cdot x_2)$ |
| 4   | 6    | 2  | 43   | -45        | $x_4 = x_2 - (q_4 \cdot x_3)$ |
| 5   | 3    | 3  | -150 | <b>157</b> | $x_5 = x_3 - (q_5 \cdot x_4)$ |
| 6   | 0    | 2  | -    | -          | Berechnung von y äquivalent.  |
| => $1029 \cdot 157 \equiv 1 \pmod{1077}$ und $-150 \cdot 1077 + 157 \cdot 1029 = 3$ |      |    |      |            |                               |

## Fibonacci-Zahlen

Rekursive Definition:

$$f_0 = 0, \quad f_1 = 1 \quad \text{und} \quad f_n = f_{n-1} + f_{n-2}$$

### Fibonacci und ggT:

$$\Rightarrow f_{n+2} \pmod{f_{n+1}} = f_n$$

⇒ ggT zweier aufeinander folgender Fibonacci-Zahlen ist 1.

## Die Eulersche $\varphi$ -Funktion

### Definition 3.3 (Eulersche $\varphi$ -Funktion)

Gibt für eine Zahl  $n$  die Mächtigkeit an, wie viele Zahlen  $a \in [0, \dots, n-1]$  kein Teiler von  $n$  sind:  $\Phi(n) = \#\{a : 0 \leq a \leq n-1, ggT(a, n) = 1\}$

⇒  $n \in \mathbb{N}$  ist genau dann eine Primzahl, wenn  $\varphi(n) = n-1$  gilt.

### Satz 3.4 (Berechnung Eulersche $\varphi$ -Funktion)

Für  $n = p_1^{e_1} \dots p_r^{e_r}$  mit verschiedenen Primzahlen  $p_i$  und  $e_i \geq 1$  gilt:

$$\varphi(n) = \prod_i p_i^{e_i-1} (p_i - 1)$$

## Kongruenzrechnung

### Definition 3.5

Für  $a, b \in \mathbb{Z}$  sagt man,  $a$  ist kongruent  $b$  modulo  $m$ , in Zeichen  $a \equiv b \pmod m$ , falls  $m | a - b$  gilt.

### Invertierbarkeit modulo $n$

### Definition 3.6

Für  $n \in \mathbb{N}$  und  $a \in \mathbb{Z}$  ist  $a$  invertierbar modulo  $n$ , falls ein  $b \in \mathbb{Z}$  existiert mit  $ab \equiv 1 \pmod n$ .

⇒ **a ist genau dann invertierbar modulo  $n$ , wenn  $ggT(n, a) = 1$  ist.**

⇒ Ist  $a$  invertierbar, so findet man mit dem erweiterten euklidischen Algorithmus  $x, y \in \mathbb{Z}$  mit  $xn + ya = 1$ . **Somit ist  $y$  invers zu  $a$  modulo  $n$**

### Modulo-Gleichungen lösen

### Satz 3.7

Seien  $a, b \in \mathbb{Z}$  und  $m \in \mathbb{N}$ . Dann gilt:

1. Die Gleichung  $ax \equiv b \pmod m$  ist **genau dann lösbar**, wenn  $ggT(a, m) | b$  gilt.
2. Gilt  $ggT(a, m) | b$ , so gibt es genau  $ggT(a, m) \pmod m$  verschiedene Lösungen der Gleichung  $ax \equiv b \pmod m$ .

### Repräsentantensystem

- Menge aller Zahlen, die eine Modulgleichung lösen.
- Es müssen nur alle **Zahlen kleiner als der Modulator** betrachtet werden.
- Mit zwei Unbekannten: Modulgleichungen aufaddieren und eine Unbekannte eliminieren.
- Beispiel:  $6x \equiv 12 \pmod{36} \Rightarrow x \in [\bar{2}; \bar{8}; \bar{14}; \bar{20}; \bar{26}; \bar{32}]$

### Der chinesische Restsatz

Seien  $m_r \in \mathbb{N}$  mit  $ggT(m_i, m_j) = 1 \quad \forall i \neq j$  und  $a_r \in \mathbb{Z}$  gegeben. Dann ist das Kongruenzgleichungssystem mit genau einer Lösung für  $x \pmod{\prod_i m_i}$  lösbar:

$$x \equiv a_1 \pmod{m_1} \tag{1}$$

$$x \equiv a_2 \pmod{m_2} \tag{2}$$

$$\dots \tag{3}$$

$$x \equiv a_r \pmod{m_r} \tag{4}$$

Lösung:

1. Tipp: Modulatoren verkleinern mit Primfaktorzerlegung.
2.  $M_i$  **berechnen:** Produkt aller  $m_l$  außer  $m_i$ .
3. Erweiterter Euklid von  $M_i$  und  $m_i$ : Lösung  $z_i$  ( $z_i$  eig.  $x$ ).
4. Einsetzen:  $x = a_1 \cdot z_1 \cdot M_1 + \dots + a_r \cdot z_r \cdot M_r$
5. **Ergebnis verkleinern:**  $x \pmod{\prod_i m_i}$

### Satz 3.8 (Spezialfall: 2 Gleichungen)

Seien  $m_1, m_2 \in \mathbb{N}$  mit  $ggT(m_1, m_2) = 1$  und  $a_1, a_2 \in \mathbb{Z}$ . Bestimme mit dem erweiterten euklidischen Algorithmus  $x, y \in \mathbb{Z}$  mit  $xm_2 + ym_1 = 1$ . Dann löst  $a = (a_1 m_2 x + a_2 m_1 y) \pmod{m_1 m_2}$  das Kongruenzgleichungssystem

$$x \equiv a_1 \pmod{m_1}, \quad x \equiv a_2 \pmod{m_2}.$$

## 4 Primzahltests

### Lemma 4.1

Für eine Primzahl  $p$  und ganzen Zahlen  $a, b$  gilt  $(a + b)^p \equiv a^p + b^p \pmod p$ .

### Satz 4.2 (Kleiner Satz von Fermat)

Für eine Primzahl  $p$  und eine ganze Zahl  $a$  gelten die Aussagen  $a^p \equiv a \pmod p$  und  $\text{ggT}(a, p) = 1 \Rightarrow a^{p-1} \equiv 1 \pmod p$ .

⇒ Für die ersten 200 Zahlen gilt auch die Umkehrung des Satzes!

### Satz 4.3 (Satz von Euler)

Ist  $n \in \mathbb{N}$  und  $a \in \mathbb{Z}$  mit  $\text{ggT}(a, n) = 1$ , so gilt

$$a^{\varphi(n)} \equiv 1 \pmod n.$$

### Satz 4.4 (Potenzen modulo $n$ lösen)

$k, n \in \mathbb{N}$  und  $a \in \mathbb{Z}$  mit  $\text{ggT}(a, n) = 1$  :

$$a^k \pmod n \equiv a^{k \pmod{\varphi(n)}} \pmod n$$

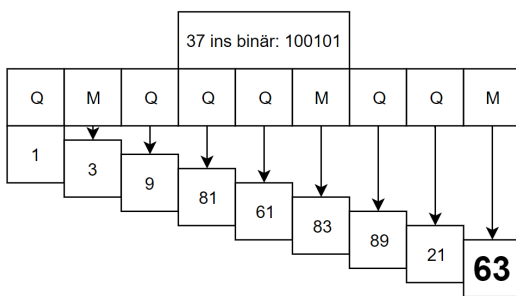
Beispiel:

$$333^{33^3} \pmod 8 \Rightarrow \text{ggT}(333, 8) = 1 \Rightarrow \varphi(8) = 4 \Rightarrow 33^3 \pmod 4 = 1 \\ \Rightarrow 333^{35937} \pmod 8 = 333^1 \pmod 8 = 5$$

### Die square-and-multiply-Methode

- Gegeben:  $a^b \pmod n$  mit Zwischenergebnis  $x \equiv 1$ .
- Definition:  $Q \equiv x^2, M \equiv x \cdot a$ .
- $b_{(10)} \rightarrow b_{(2)}$  (Binärumschreibung des Exponenten).
- $0 \in b \rightarrow Q$  bzw.  $1 \in b \rightarrow QM$ .
- Sequentielle Berechnung mit modulation nach jedem Schritt. Nicht vergessen, dass  $x$  mit 1 initialisiert wird!**

Beispiel  $3^{37} \pmod{100}$ :



### Der Fermatsche Primzahltest

#### Definition 4.5

Eine zusammengesetzte natürliche Zahl  $n$  heißt Fermat-Pseudoprimzahl zur Basis  $a$ , wenn  $\text{ggT}(n, a) = 1$  gilt und  $a^{n-1} \equiv 1 \pmod n$ .

⇒ Fermat-Pseudoprimzahlen erfüllen also den Primzahltest, obwohl sie keine Primzahlen sind.

Test: Berechne mit der square-and-multiply-Methode  $b = a^{n-1} \pmod n$ . Ist  $b = 1$ , so ist  $n$  eine Primzahl oder eine Fermat-Pseudoprimzahl zur Basis  $a$ .

## Carmichael-Zahlen

### Definition 4.6

Eine zusammengesetzte natürliche Zahl  $n$  heißt Carmichael-Zahl, wenn für alle natürlichen Zahlen  $a$  mit  $\text{ggT}(a, n) = 1$  gilt  $a^{n-1} \equiv 1 \pmod n$ .

### Satz 4.7 (Korselt-Kriterium)

Eine natürliche Zahl  $n$  ist genau dann eine Carmichael-Zahl, wenn die Primfaktorzerlegung von  $n$  die Gestalt  $n = p_1 \dots p_r$  mit  $r \geq 3$  hat und  $p_i - 1 \mid n - 1$  für  $1 \leq i \leq r$  gilt.

### Der Miller-Rabin-Test Algorithmus zum Miller-Rabin-Test:

Gegeben:  $n$  mit  $n \pmod 2 \neq 0$ :

- $x = n - 1 \Rightarrow x = 2^l \cdot \text{Rest}$
- if ( $a^{\text{Rest}} \pmod n == 1$ ) true
- else if ( $\exists i \in [0, l - 1] : a^{\text{Rest} \cdot 2^i} \pmod n == -1$ ) true
- else false

## 5 Public-Key-Verschlüsselung

### RSA-Verfahren

- Key: Man wählt Primzahlen  $p, q > 2$  mit  $p \neq q$  und berechnet  $N = pq$ . Wähle eine natürliche Zahl  $e > 1$  mit  $\text{ggT}(e, (p - 1), (q - 1)) = 1$  und berechne  $ed \equiv 1 \pmod{(p - 1)(q - 1)}$ . Der öffentliche Schlüssel ist  $(N, d)$ , der private  $(N, e)$ .  
Effiziente Berechnung von  $e$ :  **$e$  ist die erste Primzahl, die nicht in der Primfaktorzerlegung von  $(p - 1) \cdot (q - 1)$  enthalten ist.**
- Verschlüsselung: Nachricht wird mit öffentlichem Schlüssel der Person über die square-and-multiply-Methode verschlüsselt, die die Nachricht erhalten soll. Hierbei wird der Text in Blöcke  $a$  unterteilt.  
→ Funktion:  $\text{Chiffre} = a^e \pmod N$  für alle Blöcke  $a$ .
- Entschlüsselung: Mit dem privaten Schlüssel kann die Nachricht über die square-and-multiply-Methode entschlüsselt werden.  
→  $a = \text{Chiffre}^d \pmod N$  für alle Chiffreblöcke.

### Faktorisieren von $N$ mittels Quadraten

- Haben wir eine Gleichung  $x^2 \equiv 1 \pmod N$  gegeben, so hat diese vier Lösungen:  $1, N-1, w, N-w$   
→  $p = \text{ggT}(N, w-1)$  und  $q = \frac{N}{p}$ .

### Die Fermatsche Faktorisierungsmethode

- Gegeben: Eine RSA-Zahl  $N$ . Gesucht: **Zwei ungerade Primzahlen  $p < q$ .**
- Berechne  $u = \lfloor \sqrt{N} \rfloor$
- $u = u + 1, v = u^2 - N, w = \sqrt{v}$
- if ( $w \in \mathbb{Z}$ ):  $p = u - w, q = u + w$ , else goto 3.

## Kettenbruchzerlegung einer Rationalen Zahl

Führe den euklidischen Algorithmus auf die rationale Zahl  $a$  durch. Die **Multiplikatoren**  $q$  bilden als Vektor zusammengestellt die Kettenbruchzerlegung.

Beispiel:

Kettenbruchentwicklung Fibonacci  $\frac{f_{n+3}}{f_{n+2}}$  hat erst  $n$  Einsen und dann eine 2.

### Näherungsbrüche

Näherungsbruch aus Kettenbruchentwicklung bestimmen:

1. Seien  $a_0, a_1, \dots, a_m$  gegeben.
2. Initiiere  $p_{-2} = 0, p_{-1} = 1, q_{-2} = 1, q_{-1} = 0$ .
3. Berechne Rekursiv  $p_n = a_n p_{n-1} + p_{n-2}, q_n = a_n q_{n-1} + q_{n-2}$  für  $n \geq 0$ .
4. **n-ter Näherungsbruch:**  $p_n/q_n$ .

## 6 Die Pollardsche $\rho$ -Methode zur Faktorisierung

### Berechnen von Vorperiode, Periode und Periodenlänge

- Gegeben: Rekursiv definierte Modulo-Gleichung.
- Vorperiode: Alle Elemente, die sich **nicht** periodisch wiederholen.
- Periode: Elemente, die sich **periodisch** wiederholen.
- Vorperiodenlänge  $s : s = \#Vorperiode$
- Periodenlänge  $t : t = \#Periode$
- **Kleinster Index  $l \leq 1$ , für den  $x_l = x_{2l}$  gilt:**

$$l = \begin{cases} t & \text{für } s = 0, \\ \lceil \frac{s}{t} \rceil t & \text{für } s > 0. \end{cases}$$

### Faktorisierung mit der Pollardschen $\rho$ -Methode:

Sei  $n$  eine zusammengesetzte natürliche Zahl ohne kleine Teiler.

1. Wähle  $x, a \in \mathbb{Z}$ , setze  $y = x$ .
2. Berechne  $x = x^2 + a \bmod n, y = y^2 + a \bmod n, y = y^2 + a \bmod n$ .
3. Berechne  $d = ggT(x - y, n)$ . Falls  $d = 1$ , gehe zu (2).
4. Gilt  $1 < d < n$  gib  $d$  als nichttrivialen Teiler von  $n$  aus, andernfalls gehe zu (2).

### Finden einer Zahl $n$ mit bestimmten Teiler

Aufgabe: Finde einen Primfaktor  $d$  der Zahl  $n$ , sodass die Pollardsche  $\rho$ -Methode nach  $k$  Schritten terminiert.

Gegeben: Anfangswerte  $x, a$

1. Wende  $k-1$  - mal die Pollardsche  $\rho$ -Methode an mit der Annahme, das  $d = ggT(x - y, n) = 1$ .
2.  $k$ -te Durchführung: Berechne Primfaktorzerlegung von  $x - y$  und gebe den größten Primfaktor als  $d$  zurück.
3. Ergebnis sind alle Vielfachen von  $d$ .

## 7 Kryptographische Anwendung diskreter Logarithmen

### Die Ordnung

- Ist  $n \in \mathbb{N}, a \in \mathbb{Z}$  mit  $ggT(n, a) = 1$ , so gilt  $a^{\varphi(n)} \equiv 1 \bmod n$ .
- Die Ordnung  $ord_n(a)$  von  $a$  modulo  $n$  ist der erste Exponent  $k$ , für den gilt:

$$a^k \equiv 1 \bmod n$$

### Eigenschaften der Ordnung

Sei  $n \in \mathbb{N}$  und  $a \in \mathbb{Z}$  mit  $ggT(n, a) = 1$ . Für  $k, k_1, k_2 \in \mathbb{N}_0$  gilt dann:

1.  $a^k \equiv 1 \bmod n \Leftrightarrow ord_n(a) | k$ .
2.  $ord_n(a) | \varphi(n)$ .
3.  $a^{k_1} \equiv a^{k_2} \bmod n \Leftrightarrow k_1 \equiv k_2 \bmod ord_n(a)$ .
4. Für  $k \in \mathbb{N}$  gilt  $ord_n(a^k) = \frac{ord_n(a)}{ggT(ord_n(a), k)}$ .
5.  $ord_n(p_1, p_2) = kgV(ord_n(p_1), ord_n(p_2))$  mit  $p$  Primzahlen.
6. Gilt  $ord_p(a) = p - 1$ , so heißt  $a$  Primitivwurzel.

### Bestimmung von $ord_n(a)$

Naive Methode:

1. Sei  $n \in \mathbb{N}, a \in \mathbb{Z}$  mit  $ggT(n, a) = 1$ .
2. Für  $k = 1, \dots, n$ : Gebe  $k$  als  $ord_n(a)$  zurück, falls  $a^k \equiv 1 \bmod n$ .

Allgemeine Methode:

1. Sei  $n, m \in \mathbb{N}, a \in \mathbb{Z}$  mit  $a^m \equiv 1 \bmod n, q_1, \dots, q_r$  seien alle Primteiler von  $m$ .
2. Für  $i = 1, \dots, r$ : Während  $q_i | m$  und  $a^{\frac{m}{q_i}} \equiv 1 \bmod n$  gilt, setze  $m = \frac{m}{q_i}$ .
3. Es gilt  $ord_n(a) = m$ .

## 8 Hash-Funktionen

### Definition 8.1 (Hash-Funktion)

Funktionen, die beliebig lange Zeichenketten in Zeichenketten einer fest vorgegebenen Länge  $n$  umwandeln.

### Eigenschaften einer kryptographischen Hash-Funktion

- Für jede mögliche Zeichenkette soll der Hashwert  $h(a)$  schnell und effektiv berechenbar sein.
- $h$  soll eine Einwegfunktion sein, d.h. mit der Umkehrfunktion von  $h$  kann das entsprechende  $a$  nicht gefunden werden.
- $h$  ist stark kollisionsresistent, d.h. man kann praktisch kein  $a_2$  finden, sodass  $h(a) = h(a_2)$ .

### Anwendungen

- Test, ob erhaltene Nachrichten verändert wurde oder nicht:  
A sendet  $a$  und  $h(a)$  an B, B erhält  $a_2$ , berechnet  $h(a_2)$ . Ist  $h(a) = h(a_2)$ , so kann wegen der Kollisionsresistenz "nicht verändert" geschlossen werden.
- Beispiel: Verschlüsselung der Passwörter unter Unix

## 9 Digitale Signaturen

### RSA-Signatur

1. Key: Öffentlicher und privater RSA-Schlüssel und Hash-Funktion H.
2. Signatur: Berechne Hash-Wert  $h = H(M)$  einer Nachricht M und berechne mit privatem Schlüssel die RSA-Signatur s:

$$s = h^d \pmod{N}$$

3. Signaturüberprüfung: Mit öffentlichem Schlüssel prüfen ob gilt:

$$H(M) \equiv s^e \pmod{N}$$

### Die ElGamal-Signatur

1. Key:
  - (a) Eine große Primzahl p, dazu eine Primitivwurzel  $g \pmod{p}$  mit  $3 \leq g \leq p-2$  und g ist kein Teiler von  $p-1$ .
  - (b) Wahl einer geheimen Zahl  $e$  mit  $2 \leq e \leq p-2$  und  $ggT(e, p-1) = 1$ .
  - (c) Berechnen von  $f = g^e \pmod{p}$ .  
→ Öffentlicher Schlüssel: Tripel (p,g,f)  
→ Privater Schlüssel: e
  - (d) Festlegung einer kryptographischen Hash-Funktion.
2. Signatur: Bestimmen des Hash-Wertes h, Wahl einer zufälligen Zahl z mit  $1 \leq z \leq p-2$ ,  $ggT(z, p-1) = 1$  und berechnen von

$$b = g^z \pmod{p}$$

$$c = \frac{1}{z}(h - be) \pmod{p-1}$$

→ Signatur ist das Paar (b,c).

3. Signaturüberprüfung: Berechnen des Hash-Wertes h, überprüfen von  $1 \leq b \leq p-1$  und  $g^h \equiv f^b \cdot b^c \pmod{p}$ .
4. Angriff:
  - (a) Logarithmenberechnung aus  $g^e \equiv f \pmod{p}$  zur Berechnung des privaten Exponenten e.
  - (b) Ohne Verschlüsselung sind evtl. h und (b,c) zugänglich, dann gilt:

$$h \equiv b \cdot e + c \cdot z \pmod{p-1}$$

→ z muss geheim gehalten werden, ansonsten ist Gleichung lösbar.

### Verschlüsselung mit ElGamal

1. Klartext a wird mit dem öffentlichen ElGamal-Schlüssel (p,g,f) des Empfängers und mindestens einer Zufallszahl z verschlüsselt zu:  $c = a \cdot f^z \pmod{p}$
2. Zudem muss noch  $b = g^z \pmod{p}$  berechnet werden.

### Entschlüsselung ElGamal

Gegeben:  $a_i = c_i \cdot b_i^{p-e-1} \pmod{p}$

**Variante 1:** Square and Multiply auf  $b_i^{p-e-1}$ .

**Variante 2:** Exponenten von b halbieren durch  $b^2 \pmod{p} \dots b^2 \pmod{p}$ . Falls x ungerade:  $b \cdot b^{Exponent-1}$ .

### Diffie-Hellman-Schlüsselaustausch

1. Grundlage ist die ElGamal-Verschlüsselung mit Schlüsseln  $(g, p_1, e_1), (g, p_2, e_2)$ .
2. Daraus können die  $f_i$  wie folgt bestimmt werden:  $f_i = g^{e_i} \pmod{p_i}$
3. Der gemeinsame Schlüssel wird wie folgt bestimmt (aus Sicht von 1):

$$k_{1,2} = g^{e_1 e_2} \pmod{p} = f_2^{e_1} \pmod{p}$$

## 10 Berechnung diskreter Algorithmen

Gegebene Gleichung:  $g^x \equiv a \pmod{p}$

Gesucht: Lösung für  $x = \log_g a$

### Naive Methode

1. Setze  $g_0 = 1$  und berechne rekursiv  $g_n = g_{n-1}g \pmod{p}$  für  $n = 1, 2, \dots$ . Für jedes n testet man:
2. Gilt  $g_n = a$ , so ist n der gesuchte diskrete Logarithmus.
3. Gilt  $g_n = 1$ , so gibt es keine Lösung.

### Pollards $\rho$ -Methode

1. Annahme: g ist Primitivwurzel von p und es wurden bereits Exponenten s,t gefunden, die  $a^s \equiv g^t \pmod{p}$  lösen.
2. Dann kann die Gleichung  $sx \equiv t \pmod{p-1}$  gelöst werden. Diese Gleichung hat  $d = ggT(s, p-1)$  verschiedene Lösungen, die durchprobiert werden müssen.