Kryptographie I Zusammenfassung

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: Häufgkeitsanalyse 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. Ausgelesen werden alle Spalten von links nach rechts, oben nach unten
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: Aufüllzeichen betrachten oder Schlüssellänge raten

ALBC-1-Verschlüsselung

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

ALBC-k-Verschlüsselung

1. key: $x_1, \ldots, x_{k^2+k}$ mit $ggT(detA, N)$ (N = Alphabetlänge , $A = k \times k$ -Matrix)
2. Verschlüsselungsfunktion: $f(y_1, \ldots, y_k) = Ay + (x_{k^2+1}, \ldots, x_{k^2+k}) \mod N$
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: $b_i = b_{i-1} + a_i + k \mod 26$

Definition 1.1 (*Padding*)

Auffüllen des Ausgangstextes, sodass er bei Blockchiffrierungen durch die Länge k teilbar ist.

STROM-Chiffrierung

1. Es werden nur Großbuchstaben berücksichtigt!
2. Der Schlüssel besteht aus einer Folge von Großbuchstaben $k_1, k_2 \ldots$, die mindestens so lang ist wie der verschlüsselte Text.
3. Verschlüsselungsfunktion: $f(a_i, k_i) = a_i + k_i \mod 26$
4. Entschlüsselungsfunktion: $g(b_i, k_i) = b_i - k_i \mod 26$
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

1. Schlüsselwort $k_1, k_2 \ldots, k_n$, welches periodisch auf den Text angewandt wird.
2. Verschlüsselungsfunktion: $b_i = a_i + k_i \ mod\ 26$.
3. Entschlüsselungsfunktion: $a_i = b_i - k_i \ mod\ 26$.
4. Angriff: Kasiski-Test.
5. "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

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

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

   Bei entsprechend vielen gleichen kurzen Zeichenketten kann man hoffen, dass

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

   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

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

Homophone Substitutionschiffrierung

1. key: 10×10 -Tabellenmatrix mit Buchstaben
2. Verschlüsselung: Buchstaben suchen und zu Spaltenzahl und Zeilenzahl ändern
3. Entschlüsselung: Umkehrfunktion
4. Angriff: Tabelle anlegen, Wörter erraten, Buchstaben damit eintragen und damit weiter raten.
5. Verbesserung: Abhängig von der Häufigkeit der Buchstaben Tabelle befüllen, d.h. mehr E als X

Beispiel: Erlangen → 22 03 28 42 61 33 22 65

	1	2	3	4	5	6	7	8	9	0
1	I	P	I		O	U	O		P	N
2	W	E	U	T	E	K		L	O	
3	E	U	G	N	B	T	N		S	T
4	T	A	Z	M	D		I	O	E	
5	S	V	T	J		E		Y		H
6	N	A	O	L	N	S	U	G	O	E
7		C	B	A	F	R	S		I	R
8	I	C	W	Y	R	U	A	M		
9	M	V	T		H	P	D	I	X	Q
0	L	S	R	E	T	D	E	A	H	E

Drehraster-Chiffrierung für 6×6-Schablonen

1. key: Schablone mit 9 Löchern abhängig von Bahnmatrix
2. Konstruktion Bahnmatrix: Felder (i, j), wobei $1 \leq i, j \leq 6$. Durch Drehung vom (i, j) erhält man $(j, 7 - i), (7 - i, 7 - j), (7 - j, i)$ als eine Bahn.
3. Verschlüsselung: Aufteilung des Ausgangstextes in Blöcke der Länge 36. Eintragen des Textes in Schablone mit 90° Drehungen im Uhrzeigersinn. Für jeden Block wiederholen.
4. Entschlüsselung: Blöcke der Länge 36 in Matrizen eintragen und mit Schablone entschlüsseln.
5. Angriff: Erste 36 Ziffern in Matrix. Buchstaben in Matrix vermuten und mit Bahnmatrix 3 Felder ausschließen. Verfahren mit jedem "bekannten" Buchstaben wiederholen.

ADFGVX-Chiffrierung

1. Das Klartextalphabet besteht aus 36 Zeichen, den Buchstaben $A, \ldots, Z$ und den Ziffern $0, 1, \ldots, 9$. Das Chiffretextalphabet aus den Zeichen A, D, F, G, V, X.
2. Schlüssel: Erstellen einer 6×6 und Wahl eines Permutationsschlüssels der Länge n.
3. 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.
4. 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

$$a \bmod b = a - \lfloor \frac{a}{b} \rfloor \, b.$$

Tipp: modulo im Taschenrechner:
A / B = X,Y → X,Y - X = 0,Y → 0,Y · B = Rest

Definition 3.2 (*Teilt-Operator*)

Für $a, b \in \mathbb{Z}$ sagt man 'a teilt b' und schreibt $a|b$, falls ein $c \in \mathbb{Z}$ existiert mit $b = ca$.

Satz 3.3 (*Fundamentalsatz der Arithmetik*)

Jede ganze Zahl $n \neq 0$ lässt sich eindeutig schreiben als

$$n = \pm p_1^{e_1} p_2^{e_2} \dots p_r^{e_r}$$

mit paarweise verschiedenen Primzahlen $p_1, \dots, p_r$ und $e_i \in \mathbb{N}$.

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: $n \mod p_i = 0 \Rightarrow p_i$ kleinster Teiler.
→ Wenn kein p_i existiert ⇒ n ist eine Primzahl

Größter gemeinsamer Teiler

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

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

Seien $m, n \in \mathbb{Z}$. Dann hat $d = ggT(m,n)$ die folgenden Eigenschaften und ist dadurch eindeutig bestimmt:

1. $d \in gT(m,n)$
2. $d' \in gT(m,n) \Rightarrow d'|d$
3. $d \geq 0$

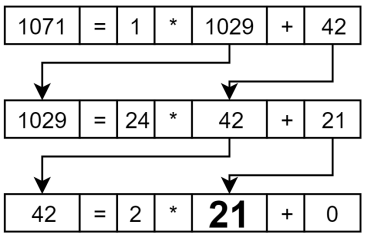
Kleinstes gemeinsames Vielfaches Die Menge $gV(m,n)$ ist die Menge der gemeinsamen Vielfachen von $m, n \in \mathbb{Z}$. Das kleinste Element dieser Menge nennt man kleinstes gemeinsame Vielfache $kgV(m,n)$.

Nützliche Tipps:
→ $m, n \in \mathbb{Z}$ sind teilerfremd, wenn $ggT(m,n) = 1$ gilt.
→ ggT(m,n) = Multiplikation niedrigster Potenzen gemeinsamer Primfaktoren
→ kgV(m,n) = Multiplikation höchter Potenzen jeglicher Primfaktoren
→ Pimfaktoren 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 3.4

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 definiert die rekursiven Zahlenfolgen $a_i, b_i, x_i, x_i', y_i, y_i'$ wie folgt:

- $a_0 = a,\ b_0 = b,\ x_0 = 1,\ x_0' = 0,\ y_0 = 0,\ y_0' = 1.$
- Ist $b_i = 0$, so sind wir fertig.
  Ist $b_i > 0$, so definiert man $q_i = \lfloor \frac{a_i}{b_i} \rfloor$ und damit
  $a_{i+1} = b_i,\ b_{i+1} = a_i - q_i b_i,\ x_{i+1} = x_i',$
  $x_{i+1}' = x_i - q_i x_i',\ y_{i+1} = y_i',\ y_{i+1}' = y_i - q_i y_i'$

Ist $n \geq 0$ der Index mit $b_n = 0$, so gilt

$$ggT(a,b) = a_n \text{ und } ggT(a,b) = x_n a + y_n b$$

Beispiel:

Zeile	a	q	x	y	Berechnung
0	1077	-	1	0	Vorinitialisiert
1	1029	-	0	1	Vorinitialisiert
2	48	1	1	-1	x2 = x0 - (q2*x1)
3	21	21	-21	22	x3 = x1 - (q3*x2)
4	6	2	43	-45	x4 = x2 - (q4*x3)
5	3	3	-150	**157**	x5 = x3 - (q5*x4)
6	0	2	-	-	Berechnung von y äquivalent
=> 1029 * 157 <=> 1 mod 1077 und -150*1077 + 157*1029 = 3					

Fibonacci-Zahlen
Rekursive Definition:

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

Fibonacci und ggT: $\Rightarrow f_{n+2} \mod f_{n+1} = f_n \Rightarrow$ ggT zweier aufeinander folgender Fibonacci-Zahlen ist 1.

Die Eulersche φ-Funktion

Definition 3.5

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

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

Satz 3.6

Für die eulersche Φ-Funktion gilt:

1. *Sind $m, n \in \mathbb{N}$, so gilt $ggT(m,n) = 1 \Rightarrow \varphi(mn) = \varphi(m)\varphi(n)$.*
2. *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 3.7

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

$\Rightarrow$ Durch $a \equiv b \bmod m$ wird eine Äquivalenzrelation definiert, die Äquivalenzklasse wird mit $\bar{a}$ bezeichnet.

Lemma 3.8

Seien $a, b \in \mathbb{Z}$ und $d, m \in \mathbb{N}$.

1. *Gilt $a \equiv b \bmod m$ und $d|m$, so gilt auch $a \equiv b \bmod d$.*
2. *Gilt $a \equiv b \bmod m$, $a \equiv b \bmod n$ und $ggT(m,n) = 1$, so folgt $a \equiv b \bmod mn$.*

Invertierbarkeit modulo n

Definition 3.9

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

$\Rightarrow$ a ist genau dann invertierbar modulo n, wenn $ggT(n,a) = 1$ ist.
$\Rightarrow$ 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 3.10

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

1. *Die Gleichung $ax \equiv b \bmod m$ ist genau dann lösbar, wenn $ggT(a,m)|b$ gilt.*
2. *Gilt $ggT(a,m)|b$, so gibt es genau $ggT(a,m) \bmod m$ verschiedene Lösungen der Gleichung $ax \equiv b \bmod m$.*
3. *Gleiche Vielfache von a,b,m werden durch das gleiche Repräsentantensystem gelöst.*
4. *Gilt $ggT(a,m) = 1$ und ist $\tilde{a}$ invers zu a modulo m, so hat man die Äquivalenz:*

$$ax \equiv b \bmod m \Leftrightarrow x \equiv \tilde{a}b \bmod m.$$

Repräsentantensystem $\rightarrow$ Menge aller Zahlen, die eine Modulogleichung lösen.
$\rightarrow$ Es müssen nur alle Zahlen kleiner als der Modulator betrachtet werden.
$\rightarrow$ Beispiel: $6x \equiv 12 \mod 36 \Rightarrow x \in [\bar{2}; \bar{8}; \bar{14}; \bar{20}; \bar{26}; \bar{32}]$

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

$$x \equiv a_1 \bmod m_1 \quad (1)$$
$$x \equiv a_2 \bmod m_2 \quad (2)$$
$$\dots \quad (3)$$
$$x \equiv a_r \bmod m_r \quad (4)$$

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:$x \mod \prod_i m_i$

Satz 3.11 (*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 $a = (a_1 m_2 x + a_2 m_1 y) \bmod m_1 m_2$ das Kongruenzgleichungssystem

$$x \equiv a_1 \bmod m_1,\ x \equiv a_2 \bmod m_2.$$

Satz 3.12 (*Spezialfall: Primzahlpotenzen*)

Sei p eine Primzahl, $m, n \in \mathbb{N}$ mir $m \geq n$ und $a, b \in \mathbb{Z}$. Dann ist das Kongruenzgleichungssystem $x \equiv a \bmod p^m$ und $x \equiv b \bmod p^n$ lösbar mit $x \equiv a \bmod p^m$, falls $x \equiv a \bmod p^n$.

$\rightarrow$ Beispiel:

Algorithmus zur Berechnung von $\lfloor \sqrt{n} \rfloor$

1. Setze $x = n$.
2. Setze $y = \lfloor \frac{1}{2}(x + \lfloor \frac{n}{x} \rfloor) \rfloor$. Falls $y \geq x$, so ist x die Lösung. Ansonsten setze $x = y$ und wiederhole Schritt (2).

4 Primzahltests

Satz 4.1

Seien $p_1, \ldots, p_r$ die ersten r Primzahlen und $P = p_1 \ldots p_r$. Dann gilt:

1. *Ist N eine natürliche Zahl, dann sind unter den Zahlen $N+1, \ldots, N+P$ genau $P - \varphi(P)$ Stück durch eine der Primzahlen teilbar.*
2. *Der Anteil der natürlichen Zahlen, die durch eine der Primzahlen teilbar sind, beträgt also $1 - (\prod_{i=1}^{r} 1 - \frac{1}{p_i})$.*

Lemma 4.2

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

Satz 4.3 (*Kleiner Satz von Fermat*)

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

$\Rightarrow$ Für die ersten 200 Zahlen gilt auch die Umkehrung des Satzes!

Satz 4.4 (*Satz von Euler*)

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

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

Satz 4.5 (*Potenzen modulo n lösen*)

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

$$a^k \mod n = a^r \mod n$$

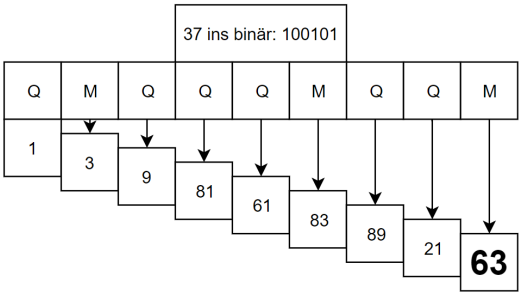
Beispiel:
$333^{33^3} \mod 8 \Rightarrow ggT(333,8) = 1 \Rightarrow \varphi(8) = 4 \Rightarrow 33^3 mod 4 = 1$
$\Rightarrow 333^{35937} \mod 8 = 333^1 \mod 8 = 5$

Letzte n Stellen von a^k berechnen:
$ggT(a, 10^n) = 1 \Rightarrow \varphi(10^n) = x \Rightarrow k \mod x = r$
$\Rightarrow a^r \mod 10^n = Ergebnis$

Die square-and-multiply-Methode

1. Gegeben: $a^b \mod n$ mit Zwischenergebnis $x \equiv 1$.
2. Definition: $Q \equiv x^2, M \equiv x \cdot a$.
3. $b_{(10)} \rightarrow b_{(2)}$ (Binärumwandlung des Exponenten).
4. $0 \in b \rightarrow$ Q bzw. $1 \in b \rightarrow$ QM.
5. Sequentielle Berechnung mit modulation nach jedem Schritt. Nicht vergessen, dass x mit 1 initialisiert wird!

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



Der Fermatsche Primzahltest

Definition 4.6

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

$\Rightarrow$ Fermat-Pseudoprimzahlen erfüllen also den Primzahltest, obwohl sie keine Primzahlen sind.
$\rightarrow$ Anzahl der Primzahlen $\leq x$: $\pi(x) \approx x/log(x)$
$\rightarrow \lim_{x \to \infty} \frac{\pi_{Fermat,a}(x)}{\pi(x)} = 0$
Test: Berechne mit der square-and-multiply-Methode $b = a^{n-1} \ mod\ n$. Ist $b = 1$, so ist n eine Primzahl oder eine Fermat-Pseudoprimzahl zur Basis a.

Carmichael-Zahlen

Definition 4.7

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

Satz 4.8 (*Korselt-Kriterium*)

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

Der Miller-Rabin-Test

Definition 4.9

Eine zusammengesetzte ungerade natürliche Zahl n heißt Miller-Rabin-Pseudoprimzahl zur Basis a oder eine starke Pseudoprimzahl zur Basis a, wenn n den Miller-Rabin-Test zur Basis a besteht.

Algorithmus zum Miller-Rabin-Test:
Gegeben: n mit$n \mod 2 \neq 0$:

1. $x = n - 1 \Rightarrow x = 2^l \cdot Rest$
2. if($a^{Rest} \mod n == 1$) true
3. else if ($\forall i \in [0, l-1] : a^{q \cdot 2^i} \mod n == -1$)true
4. else false

Beispiel:

5 Public-Key-Verschlüsselung

RSA-Verfahren
RSA-Verschlüsselung:

1. Key: Man wählt Primzahlen $p, q > 2$ mit $p \neq q$ und berechnet $N = pq$. Wähle eine natürliche Zahl $e > 1$ mit $ggT(e, (p-1), (q-1)) = 1$ und berechne $ed \equiv 1 \mod (p-1)(q-1)$. 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 $(p-1), (q-1)$ enthalten ist.

2. 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: $Chiffre = a^e \mod N$ für alle Blöcke a.

3. Entschlüsselung: Mit dem privaten Schlüssel kann die Nachricht über die square-and-multiply-Methode entschlüsselt werden. → $a = Chiffre^d \mod N$ für alle Chiffreblöcke.

Angriff auf RSA mit dem chinesischen Restsatz

1. Gegeben: $b_1 = a_i^3 \ mod \ N_1$, $b_2 = a_i^3 \ mod \ N_2$, $b_3 = a_i^3 \ mod \ N_3$
   → Privater Exponent e ist also 3!

2. Löse mit dem chinesischen Restsatz das Kongruenzgleichungssystem

$$c_i \equiv b_1 \ mod \ N_1 \tag{5}$$
$$c_i \equiv b_2 \ mod \ N_2 \tag{6}$$
$$c_i \equiv b_3 \ mod \ N_3 \tag{7}$$

3. Die Lösung ist dann $a_i = \sqrt[3]{c_i}$.

Faktorisierung einer RSA-Zahl N bei Kenntnis beider Schlüssel Verfahren:

1. Gegeben sei ein RSA-Schlüssel (N, e), (N, d).
2. Setze $k = \lfloor ed/N \rfloor$.
3. Setze $k = k + 1$.
4. Teste, ob $k | ed - 1$ gilt, wenn nicht gehe zu (3).
5. Berechne $s = N + 1 - \frac{ed-1}{k}$ und damit $\Delta = \lfloor \sqrt{s^2 - 4N} \rfloor$.
6. Ist $s^2 - 4N \neq \Delta^2$, gehe zu (3).
7. Gib $\frac{s-\Delta}{2}$ und $\frac{s+\Delta}{2}$ als Primteiler von N aus.

Die Fermatsche Faktorisierungsmethode

1. Gegeben: Eine RSA-Zahl N. Gesucht: Zwei ungerade Primzahlen $p < q$.
2. Berechne $u = \lfloor \sqrt{N} \rfloor$
3. $u = u + 1$, $v = u^2 - N$, $w = \sqrt{v}$
4. 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.
→ Kettenbruchentwicklung Fibonacci $\frac{f_{n+3}}{f_{n+2}}$ hat erst n Einsen und dann eine 2.

Näherungsbrüche
Näherungsbruch aus Kettenbruchentwicklung bestimmten:

1. Seien $a_0, a_1, \ldots, a_m$ 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.

Angriffe auf RSA mit Kettenbrüchen
Kettenbruchangriff auf RSA:

1. Gegeben hat man den öffentlichen RSA-Schlüssel (N, e), $k_0 = 0$, $d_0 = 1$.
2. Bestimme zunächst n Näherungsbrüche $\frac{k_i}{d_i}$ der Kettenbruchentwicklung von $\frac{e}{N}$.
3. Für $i = 1, \ldots, n$:
   if$(ed_i \equiv 1 \ mod \ k_i)$: $s = N + 1 - \frac{ed_i - 1}{k_i}$ und $D = s^2 - 4N$
   if$(s > 0, \sqrt{D} \in \mathbb{N}$: $p = \frac{s-\sqrt{D}}{2}, q = \frac{s+\sqrt{D}}{2}$.
4. Die Primteiler sind nun p und q, der private Exponent ist das d_i.

6 Die Pollardsche ρ-Methode zur Faktorisierung

Idee: Sei n zusammengesetzt ohne kleine Teiler und p ein Primteiler von n. Ist x_i eine Folge von Zahlen modulo n, d.h.$x_i \in \{0, 1 \ldots, n-1\}$ und gilt für zwei Indizes $k < l$ die Beziehung $x_k \equiv x_l \ mod \ p$, so gilt $p | ggT(x_k - x_l, n)$.
⇒ Man kann also hoffen, einen nichttrivialen Teiler von n gefunden zu haben.

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 ρ-Methode:
Sei n eine zusammengesetzte natürliche Zahl ohne kleine Teiler.

1. Wähle $x, a \in \mathbb{Z}$, setze $y = x$.
2. Berechne $x = x^2 + a \ mod \ n$, $y = y^2 + a \ mod \ n$, $y = y^2 + a \ mod \ n$.
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).
   → Ist $d = n$, kann man es wie oben beschrieben machen oder die Startwerte in (1) verändern.

Finden einer Zahl n mit bestimmten Teiler Aufgabe: Finde einen Primfaktor d der Zahl n, sodass die Pollardsche ρ-Methode nach k Schritten terminiert.
Gegeben: Anfangswerte x, a

1. Wende k-1 - mal die Pollardsche ρ-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 Logaritmen

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

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

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. $a^k \equiv 1 \ mod \ n \Leftrightarrow ord_n(a) | k$.
2. $ord_n(a) | \varphi(n)$.
3. $a^{k_1} \equiv a^{k_2} \ mod \ n \Leftrightarrow k_1 \equiv k_2 \ mod \ ord_n(a)$.
4. Für $k \in \mathbb{N}$ gilt $ord_n(a^k) = \frac{ord_n(a)}{ggT(ord_n(a), k)}$.
5. Ist $d \in \mathbb{N}$ mit $d | ord_n(a)$, so hat $a^{\frac{ord_n(a)}{d}}$ die Ordnung d.

Bestimmung von $ord_n(a)$
Naive Methode:

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

Allgemeine Methode:

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

Die multiplikative Gruppe des Körpers $\mathbb{F}_p$
Für eine Primzahl p wissen wir, dass die multiplikative Gruppe des endlichen Körpers $\mathbb{F}_p$ zyklisch ist.
⇒ Jedes $g \in \mathbb{F}_p$ mit $ord_p(g) = p - 1$ nennt man Primitivwurzel.
⇒ Kann für eine Primzahl p die Zahl p-1 faktorisiert werden, so kann man Primitivwurzeln über Ordnungen bestimmen. Dies sind dann nämlich alle Teiler der Zahl p-1.

8 Hash-Funktionen

Definition 8.1 (*Hash-Funktion*)

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

Notation:
→ Σ^m ist Menge der Zeichenketten der Länge m
→ Σ^* ist die Menge aller möglichen Zeichenketten.

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 Kollisionsrestistenz "nicht verändert" geschlossen werden.
→ Verschlüsselung der Passwörter unter Unix

Parametrisierte Hash-Funktionen - MACs
→ Verwendung: Überprüfbarkeit der Authentizität einer Nachricht
→ Anwendung: Parametrisierte Hash-Funktion h_K mit Schlüssel K. Nachricht a und Hashwert $h_K(a)$ verschicken, bekommene Nachricht a_2 wird als unverändert akzeptiert, falls $h_K(a) = h_K(a_2)$ gilt.
→ Verschlüsselungsverfahren:

1. Symmetrisches Verschlüsselungsverfahren: $h_K(a) = E_K(h(a))$
2. Symmetrisches Verschlüsselungsverfahren im CBC-Modus: $h_K(a)$ wird der letzte Block von $E_K(a)$

9 Digitale Signaturen

Unterschrifft Public-Key-Verschlüsselung
→ Ohne Hash-Funktion: Verschlüssele Nachricht mit privatem Schlüssel (den nur Absender kennt).
→ Mit Hash-Funktion: Einigung auf eine Hash-Funktion h, berechnen des Hash-Wertes der Nachricht und verschlüsseln mit privatem Schlüssel. Durch senden des verschlüsselten Hash-Wertes und der Nachricht können Hash-Werte verglichen werden.
→ Zur Verschlüsselung wird die Nachricht mit der Unterschrift mit dem öffentlichen Schlüssel des Empfängers verschlüsselt.

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:
$$s = h^d \mod N$$
3. Signaturüberprüfung: Mit öffentlichem Schlüssel prüfen ob gilt:
$$H(M) \equiv s^e \mod N$$

Die ElGamal-Signatur

1. Key:
   (a) Eine große Primzahl p, dazu eine Primitivwurzel g mod 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 $f = g^e \mod p$.
   → 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
$$b = g^z \mod p$$
$$c = \frac{1}{z}(h - be) \mod p-1$$
   → Signatur ist das Paar (b,c).
3. Signaturüberprüfung: Berechnen des Hash-Wertes h, überprüfen von $1 \leq b \leq p-1$ und $g^h \equiv f^b \cdot b^c \mod p$.
4. Angriff:
   (a) Logarithmenberechnung aus $g^e \equiv f \mod p$ zur Berechnung des privaten Exponenten e.
   (b) Ohne Verschlüsselung sind evtl. h und (b,c) zugänglich, dann gilt:
$$h \equiv b \cdot e + c \cdot z \mod p-1$$
   → z muss geheim gehalten werden, ansonsten ist Gleichung lösbar.

→ Aufgrund der leichteren Berechenbarkeit von e und z sollte man vermeiden, dass $g|p-1$ gilt.

Verschlüsselung mit ElGamal

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

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: $f_i = g^{e_i} \mod p_i$
3. Der gemeinsame Schlüssel wird wie folgt bestimmt (aus Sicht von 1):
$$k_{1,2} = g^{e_1 e_2} \mod p = f_2^{e_1} \mod p$$

10 Berechnung diskreter Algorithmen

Gegebene Gleichung: $g^x \equiv a \mod p$
Gesucht: Lösung für $x = log_g a$

Naive Methode

1. Setze $g_0 = 1$ und berechne rekursiv $g_n = g_{n-1} g \ mod \ p$ für n = 1,2,.... 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 ρ-Methode

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