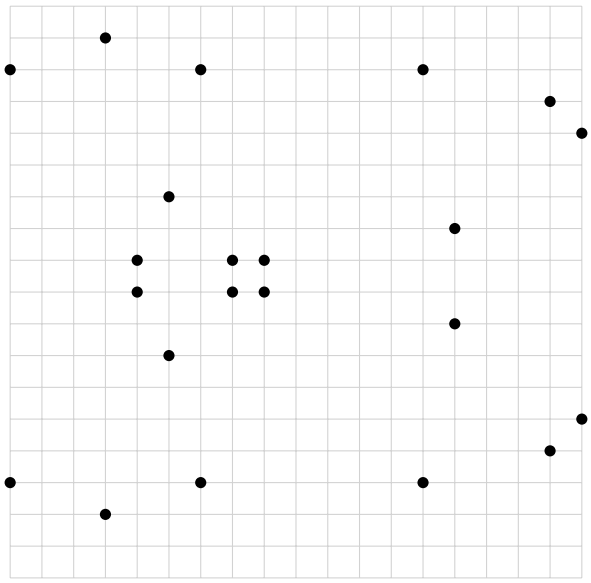
ARCHENHOLD
GYMNASIUM

Kryptologie

Begleitskript zum Informatikunterricht

Klassische Verfahren · modulares Rechnen · Diffie-Hellman · RSA · elliptische Kurven



Die 22 Punkte der Kurve $y^2 = x^3 + 2x + 9$ über $\mathbb{F}_{19}$

Inhaltsverzeichnis

1 Grundbegriffe und klassische Verfahren **2**

1.1 Worum es geht . . . 2

1.2 Caesar und die multiplikative Chiffre . . . 2

1.3 Die Vigenère-Chiffre . . . 3

1.4 Kryptoanalyse: Häufigkeiten, Kasiski und Friedman . . . 4

2 Modulare Arithmetik **6**

2.1 Rechnen modulo n . . . 6

2.2 Der euklidische Algorithmus . . . 6

2.3 Der erweiterte euklidische Algorithmus und das Inverse . . . 7

2.4 Schnelles modulares Potenzieren . . . 8

3 Diffie-Hellman-Schlüsseltausch **9**

3.1 Das Protokoll . . . 9

3.2 Sicherheit: der diskrete Logarithmus . . . 9

3.3 Primitivwurzeln . . . 10

4 RSA **11**

4.1 Die Idee: eine Falltür . . . 11

4.2 Schlüsselerzeugung . . . 11

4.3 Verschlüsseln und entschlüsseln . . . 11

4.4 Signaturen und Sicherheit . . . 12

5 Elliptische Kurven **14**

5.1 Kurven über den reellen Zahlen . . . 14

5.2 Punkte addieren . . . 14

5.3 Kurven über endlichen Körpern $\mathbb{F}_p$ . . . 15

5.4 Verschlüsseln mit elliptischen Kurven (EC-ElGamal) . . . 16

6 Überblick **17**

7 Übungen **18**

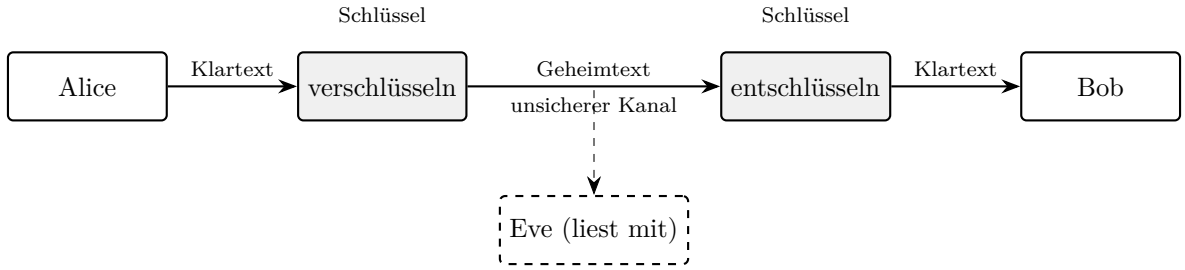
8 Lösungen **19**

1 Grundbegriffe und klassische Verfahren

So arbeiten Sie mit dem Skript: Jedes Verfahren wird erklärt, an einem Beispiel durchgerechnet und – wo es sich lohnt – als kurzer Python-Quelltext gezeigt. Die Kästen *„auf einen Blick“* fassen ein Verfahren zusammen. Abschnitt 7 enthält Übungen, Abschnitt 8 die vollständigen Lösungen: erst selbst rechnen, dann vergleichen. Alle Zahlenbeispiele sind mit einem Programm nachgerechnet.

1.1 Worum es geht

Alice möchte Bob eine Nachricht schicken. Der Übertragungsweg (Internet, Funk, Brief) ist unsicher: Eve kann mitlesen. Alice *verschlüsselt* deshalb den *Klartext* mit einem *Schlüssel* zum *Geheimtext*; Bob *entschlüsselt* ihn wieder. Kryptographie entwirft solche Verfahren, Kryptoanalyse versucht sie zu brechen – beides zusammen heißt *Kryptologie*.



Symmetrisch: Beide benutzen denselben geheimen Schlüssel (Caesar, Vigenère, AES). Problem: Wie kommt der Schlüssel sicher zu Bob? Antwort in Abschnitt 3.

Asymmetrisch: Jeder hat ein Schlüsselpaar: einen *öffentlichen* Schlüssel zum Verschlüsseln und einen *privaten* zum Entschlüsseln (RSA, elliptische Kurven, Abschnitte 4 und 5).

Kerckhoffs-Prinzip: Die Sicherheit darf nur von der Geheimhaltung des *Schlüssels* abhängen, nicht von der Geheimhaltung des Verfahrens. Alle Verfahren in diesem Skript sind öffentlich bekannt.

Schutzziele: *Vertraulichkeit* (nur Bob kann lesen), *Integrität* (die Nachricht wurde nicht verändert), *Authentizität* (sie stammt wirklich von Alice).

1.2 Caesar und die multiplikative Chiffre

Buchstaben werden zu Zahlen: `A` $= 0$, `B` $= 1$, …, `Z` $= 25$. Gerechnet wird *modulo 26* (nach `Z` geht es wieder bei `A` los, Abschnitt 2.1).

	verschlüsseln	entschlüsseln
Caesar (Verschiebung k)	$y = (x + k) \bmod 26$	$x = (y - k) \bmod 26$
multiplikativ (Faktor m)	$y = (m \cdot x) \bmod 26$	$x = (m^{-1} \cdot y) \bmod 26$

Bei Caesar gibt es nur 25 sinnvolle Schlüssel – Eve probiert einfach alle aus. Bei der multiplikativen Chiffre muss m ein *Inverses* m^{-1} besitzen ($m \cdot m^{-1} \bmod 26 = 1$), sonst landen zwei Buchstaben auf demselben Geheimtextbuchstaben. Das ist genau dann der Fall, wenn $\mathrm{ggT}(m, 26) = 1$. So kommen nur 12 Schlüssel in Frage:

m	1	3	5	7	9	11	15	17	19	21	23	25
m^{-1}	1	9	21	15	3	19	7	23	11	5	17	25

Beispiel mit $m = 7$ (Inverses $7^{-1} = 15$, denn $7 \cdot 15 = 105 = 4 \cdot 26 + 1$):

Klartext	x	$7x$	$7x \bmod 26$	Geheimtext
S	18	$7 \cdot 18 = 126$	22	W
C	2	$7 \cdot 2 = 14$	14	O
H	7	$7 \cdot 7 = 49$	23	X
U	20	$7 \cdot 20 = 140$	10	K
L	11	$7 \cdot 11 = 77$	25	Z
E	4	$7 \cdot 4 = 28$	2	C

Aus SCHULE wird WOXKZC. Wie man das Inverse ausrechnet, zeigt Abschnitt 2.3.

1.3 Die Vigenère-Chiffre

Vigenère benutzt ein *Schlüsselwort*. Der i-te Buchstabe wird um den i-ten Schlüsselbuchstaben verschoben; ist das Schlüsselwort zu Ende, beginnt es von vorn. Derselbe Klartextbuchstabe wird so je nach Position verschieden verschlüsselt – die einfache Häufigkeitsanalyse aus 1.4 funktioniert nicht mehr direkt.

$$y_i = (x_i + s_{i \bmod L}) \bmod 26, \qquad L = \text{Länge des Schlüsselworts}$$

Beispiel: Klartext HEUTE ABEND IM KINO, Schlüssel MOND (M = 12, O = 14, N = 13, D = 3).

Klartext	H	E	U	T	E	A	B	E	N	D	I	M	K	I	N	O
Schlüssel	M	O	N	D	M	O	N	D	M	O	N	D	M	O	N	D
Geheimtext	T	S	H	W	Q	O	O	H	Z	R	V	P	W	W	A	R

Geheimtext: TSHWQ OOHZR VP WWAR (Leerzeichen bleiben stehen und zählen nicht mit). Das doppelte E in HEUTE wird einmal zu S, einmal zu Q.

Quelltext 1: Vigenère verschlüsseln (Entschlüsseln: Minus statt Plus)

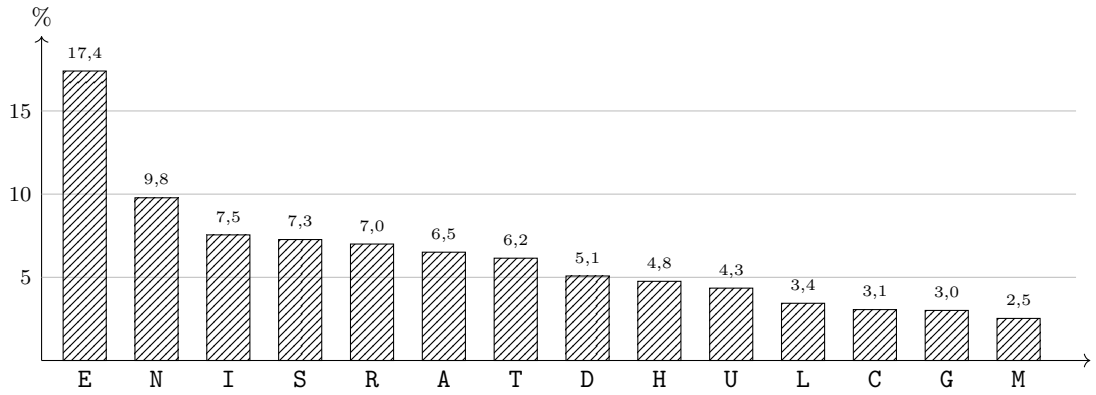
```
def vigenere_kodieren(text, schluessel):
    ergebnis = ""
    j = 0                                   # Position im Schlüssel
    for c in text:
        if c >= "A" and c <= "Z":
            x = ord(c) - 65                 # A = 0, ..., Z = 25
            s = ord(schluessel[j % len(schluessel)]) - 65
            ergebnis = ergebnis + chr((x + s) % 26 + 65)
            j = j + 1
        else:
            ergebnis = ergebnis + c         # Leerzeichen unverändert
    return ergebnis
```

Vigenère auf einen Blick	
Schlüssel	Schlüsselwort der Länge L
Rechenvorschrift	$y_i = (x_i + s_{i \bmod L}) \bmod 26$
Sicherheit beruht auf	Länge und Zufälligkeit des Schlüsselworts
Schwachstelle	Wiederholung des Schlüssels: Kasiski- und Friedman-Test finden L, danach L einfache Caesar-Analysen

1.4 Kryptoanalyse: Häufigkeiten, Kasiski und Friedman

Häufigkeitsanalyse

In deutschen Texten kommen die Buchstaben sehr ungleich oft vor. Bei Caesar wird der häufigste Geheimtextbuchstabe fast immer das `E` sein – damit ist die Verschiebung verraten.



Häufigste Buchstaben im Deutschen in Prozent. Quelle:
`https://de.wikipedia.org/wiki/Buchstabenh%C3%A4ufigkeit`

Der Kasiski-Test

Wiederholt sich im Klartext eine Buchstabenfolge und trifft sie dabei auf dieselbe Stelle des Schlüsselworts, dann wiederholt sich auch der Geheimtext. Der Abstand solcher Wiederholungen ist ein *Vielfaches der Schlüssellänge*. Ein Gauß-Zitat, mit Vigenère verschlüsselt (128 Buchstaben, Leerzeichen entfernt):

```
ETEGBEHYNLTCLTSNETEEPPNCHTNXFCWCTDEHTNHUGEEHVYDX
JPZUIWEHUSEISTECTEDCFVOYOTGCOOELNLTBFXANJVSUHEEX
FCMUUSEGBEIEFCCUSWFLJPDLJNHABFSM
```

Wiederholte Folgen aus mindestens drei Buchstaben:

Folge	Positionen	Abstand	Zerlegung	
`ETE`	1, 17	16	$2 \cdot 2 \cdot 2 \cdot 2$	
`EGBE`	3, 103	100	$2 \cdot 2 \cdot 5 \cdot 5$	
`NLT`	9, 81	72	$2 \cdot 2 \cdot 2 \cdot 3 \cdot 3$	
`HTN`	25, 36	11	11	Zufall?
`XFC`	28, 96	68	$2 \cdot 2 \cdot 17$	
`USE`	57, 101	44	$2 \cdot 2 \cdot 11$	

Alle Abstände bis auf einen sind durch 4 teilbar; der größte gemeinsame Teiler dieser Abstände ist **4**. Die Folge mit Abstand 11 ist ein *Zufallstreffer*: Zwei verschiedene Klartextstellen ergeben zufällig denselben Geheimtext. Deshalb nimmt man den ggT der *meisten* Abstände, nicht blind aller. Vermutete Schlüssellänge: $L = 4$.

Vom Schlüssel-Länge zum Schlüssel

Kennt man L, zerlegt man den Geheimtext in L Teiltexte (jeder L-te Buchstabe). Jeder Teiltext ist mit *einem* Buchstaben verschlüsselt, also eine einfache Caesar-Chiffre. Nimmt man jeweils den häufigsten Buchstaben als verschlüsseltes `E`:

Teiltext	häufigster	Anzahl	Verschiebung zu `E`	Schlüssel
1	`F`	5	1	`B`
2	`T`	6	15	`P`
3	`E`	10	0	`A`
4	`C`	6	24	`Y`

Das ergibt `BPAY` – richtig ist `BLAU`. Bei nur 32 Buchstaben je Teiltext ist das `E` nicht immer der häufigste Buchstabe. In der Praxis probiert man die zwei, drei häufigsten Kandidaten und prüft, ob beim Entschlüsseln lesbare Wörter entstehen. Je länger der Geheimtext, desto zuverlässiger klappt es.

Der Friedman-Test

Friedman schätzt die Schlüssellänge aus dem *Koinzidenzindex* I: der Wahrscheinlichkeit, dass zwei zufällig gezogene Buchstaben des Geheimtexts gleich sind ($n_A, \ldots, n_Z$ Anzahlen, n Textlänge):

$$I = \frac{\sum n_i(n_i-1)}{n(n-1)}, \qquad L \approx \frac{0{,}0377\,n}{(n-1)\,I - 0{,}0385\,n + 0{,}0762}$$

Für den Gauß-Text ergibt sich $I \approx 0,0567$ und $L \approx 2,1$. Friedman liefert also nur eine grobe Größenordnung (deutscher Klartext: $I \approx 0{,}0762$, völlig zufälliger Text: $I \approx 0{,}0385$); den genauen Wert liefert Kasiski.

One-Time-Pad: Ist der Schlüssel echt zufällig, so lang wie die Nachricht und wird nur *einmal* benutzt, gibt es keine Wiederholungen – weder Kasiski noch Friedman helfen. Diese Verschlüsselung ist nachweislich unknackbar, aber unpraktisch: Der lange Schlüssel muss vorher sicher übergeben werden.

2 Modulare Arithmetik

2.1 Rechnen modulo n

$a \bmod n$ ist der Rest bei der Division von a durch n, also eine Zahl aus $\{0, 1, \ldots, n-1\}$. Man rechnet wie auf einem Ziffernblatt: $9+5 = 14 \equiv 2 \pmod{12}$. Zwei Zahlen heißen *kongruent modulo* n ($a \equiv b \pmod n$), wenn sie bei Division durch n denselben Rest lassen.

Rechenregel: Man darf *jederzeit* zwischendurch modulo n rechnen:

$$(a+b) \bmod n = \big((a \bmod n) + (b \bmod n)\big) \bmod n, \qquad (a \cdot b) \bmod n = \big((a \bmod n) \cdot (b \bmod n)\big) \bmod n$$

Das hält die Zahlen klein – die Grundlage für das schnelle Potenzieren in 2.4.

Eine Zahl a hat modulo n ein *multiplikatives Inverses* a^{-1} mit $a \cdot a^{-1} \equiv 1 \pmod n$ genau dann, wenn $\mathrm{ggT}(a, n) = 1$. Ist $n = p$ eine Primzahl, hat also jede Zahl $1, \ldots, p-1$ ein Inverses – man kann modulo p „durch alles außer 0 teilen“.

2.2 Der euklidische Algorithmus

Den größten gemeinsamen Teiler findet man ohne Primfaktorzerlegung: $\mathrm{ggT}(a, b) = \mathrm{ggT}(b, a \bmod b)$, bis der Rest 0 ist. Beispiel $\mathrm{ggT}(1071, 462)$:

a	b	Division mit Rest
1071	462	$1071 = 2 \cdot 462 + 147$
462	147	$462 = 3 \cdot 147 + 21$
147	21	$147 = 7 \cdot 21 + 0$

Der letzte Rest ungleich 0 ist der ggT: $\mathrm{ggT}(1071, 462) = 21$.

Quelltext 2: Euklidischer Algorithmus (rekursiv)

```
1  def euklid(a, b):
       if b == 0:
           return a
       return euklid(b, a % b)
```

2.3 Der erweiterte euklidische Algorithmus und das Inverse

Darstellungssatz (Bézout): Ist $d = \text{ggT}(a, b)$, dann gibt es ganze Zahlen x, y mit

$$d = \text{ggT}(a, b) = a \cdot x + b \cdot y.$$

Diophantische Gleichung: $a \cdot x + b \cdot y = c$ ist genau dann (ganzzahlig) lösbar, wenn c ein Vielfaches von $\text{ggT}(a, b)$ ist. Für $c = \text{ggT}(a, b)$ liefert der erweiterte euklidische Algorithmus (EEA) eine konkrete Lösung x, y.

Rückwärtseinsetzen am Beispiel $\text{ggT}(128, 34)$: Aus $128 = 3 \cdot 34 + 26$, $34 = 1 \cdot 26 + 8$, $26 = 3 \cdot 8 + 2$, $8 = 4 \cdot 2 + 0$ folgt

$$2 = 26 - 3 \cdot 8 = 26 - 3 \cdot (34 - 26) = 4 \cdot 26 - 3 \cdot 34 = 4 \cdot (128 - 3 \cdot 34) - 3 \cdot 34 = 4 \cdot 128 - 15 \cdot 34.$$

Das Einsetzen wird bei größeren Zahlen unübersichtlich. Das *Tabellenschema* erledigt es systematisch.

Herleitung der Formeln. In jeder Zeile gilt $a_i = q_i \cdot b_i + r_i$, und die nächste Zeile beginnt mit $a_{i+1} = b_i$, $b_{i+1} = r_i = a_i - q_i \cdot b_i$. Steht in Zeile $i + 1$ schon $\text{ggT} = a_{i+1} \cdot x_{i+1} + b_{i+1} \cdot y_{i+1}$, dann ist

$$\text{ggT} = b_i \cdot x_{i+1} + (a_i - q_i \cdot b_i) \cdot y_{i+1} = a_i \cdot y_{i+1} + b_i \cdot (x_{i+1} - q_i \cdot y_{i+1}).$$

Der Koeffizientenvergleich mit $\text{ggT} = a_i \cdot x_i + b_i \cdot y_i$ ergibt die Formeln, mit denen man in der Tabelle *von unten nach oben* geht:

$$x_i = y_{i+1}, \qquad y_i = x_{i+1} - q_i \cdot y_{i+1}$$

Tabellenschema: Nach unten wie beim euklidischen Algorithmus dividieren ($a = q \cdot b + r$), bis der Rest $r = 0$ ist. In dieser letzten Zeile $x = 0$ und $y = 1$ setzen. Dann Zeile für Zeile nach oben: x ist das y der Zeile darunter, y wird mit der Formel berechnet (die Rechnung steht in der Tabelle).

Beispiel: $\text{ggT}(7, 26)$, gerechnet als $\text{ggT}(26, 7)$:

a	b	q	r	x	y
26	7	3	5	3	$-2 - 3 \cdot 3 = -11$
7	5	1	2	-2	$1 - 1 \cdot (-2) = 3$
5	2	2	1	1	$0 - 2 \cdot 1 = -2$
2	1	2	0	0	1

Oben steht die Bézout-Darstellung $1 = 26 \cdot 3 + 7 \cdot (-11)$.

Das Inverse ablesen: Bildet man in der Bézout-Gleichung den Rest modulo der Zahl N, so fällt $N \cdot x$ weg ($N \cdot x \bmod N = 0$). Übrig bleibt $(7 \cdot (-11)) \bmod 26 = 1$: -11 **ist das Inverse von 7 modulo 26**. Positiv gemacht: $-11 + 26 = 15$. Probe: $7 \cdot 15 = 105 = 4 \cdot 26 + 1$. ✓

Quelltext 3: Erweiterter euklidischer Algorithmus und Inverses

```
def eeuklid(a, b):
    """Gibt (d, x, y) zurück mit d = ggT(a, b) = a*x + b*y."""
    if b == 0:
        return (a, 1, 0)
    d, x, y = eeuklid(b, a % b)
    return (d, y, x - (a // b) * y)

def inverse(a, n):
    d, x, y = eeuklid(n, a)
    if d != 1:
        return None                    # kein Inverses
    return y % n
```

2.4 Schnelles modulares Potenzieren

Für Diffie-Hellman, RSA und elliptische Kurven braucht man Potenzen wie $g^a \bmod p$ mit Exponenten aus Hunderten von Stellen. a-mal zu multiplizieren dauert viel zu lange. Die Idee: *Quadrieren halbiert den Exponenten*:

$$ba^{ex} = (ba \cdot ba)^{ex/2} \quad (ex \text{ gerade}), \qquad ba^{ex} = ba \cdot ba^{ex-1} \quad (ex \text{ ungerade}).$$

Nach jedem Schritt wird modulo m gerechnet, die Zahlen bleiben klein.

Struktogramm in Worten: Start mit ba, ex, m und $pot = 1$. Solange $ex > 0$:
ex ungerade $\rightarrow$ $pot = pot \cdot ba \bmod m$, $ex = ex - 1$
ex gerade $\rightarrow$ $ba = ba \cdot ba \bmod m$, $ex = ex/2$
Am Ende steht das Ergebnis in pot.

Jede Tabellenzeile ist ein Schleifendurchlauf; die letzte Spalte entscheidet, was im nächsten Durchlauf passiert. Beispiel $6^{21} \bmod 31$:

ba	ex	pot	ex ungerade?
6	21	1	ja
6	20	6	nein
5	10	6	nein
25	5	6	ja
25	4	26	nein
5	2	26	nein
25	1	26	ja
25	0	**30**	–

Ergebnis: $6^{21} \bmod 31 = 30$ nach 7 Schritten statt 20 Multiplikationen.

Quelltext 4: Schnelles modulares Potenzieren

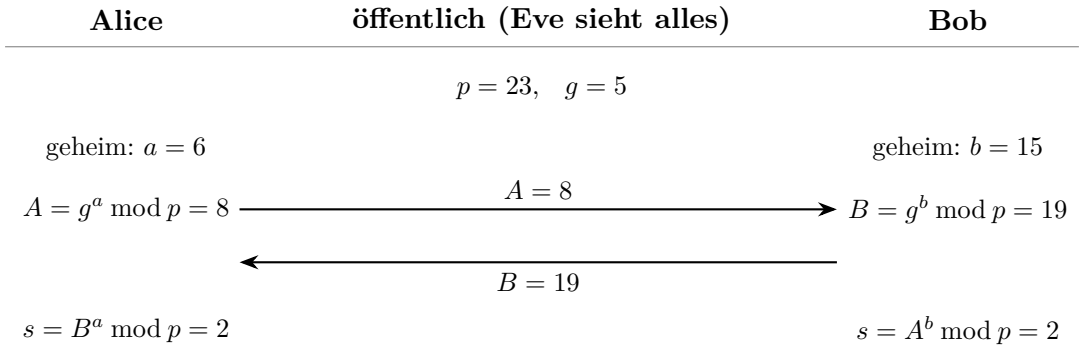
```
def modpotenz(ba, ex, m):
    pot = 1
    while ex > 0:
        if ex % 2 == 1:              # ungerade: einen Faktor abspalten
            pot = pot * ba % m
            ex = ex - 1
        else:                        # gerade: quadrieren, Exponent halbieren
            ba = ba * ba % m
            ex = ex // 2
    return pot
```

Laufzeit: Hat der Exponent k Binärstellen, braucht man höchstens $2k$ Schritte – die Laufzeit wächst nur *linear* mit der Stellenzahl. Für einen 2048-Bit-Exponenten sind das höchstens 4096 Schritte; naiv wären es über 10^{600} Multiplikationen.

3 Diffie-Hellman-Schlüsseltausch

3.1 Das Protokoll

Wie einigen sich Alice und Bob über einen abhörbaren Kanal auf einen gemeinsamen geheimen Schlüssel? Diffie und Hellman (1976) nutzen eine *Einbahnstraße*: $g^a \bmod p$ ist schnell berechnet (2.4), aber aus dem Ergebnis a zurückzugewinnen ist sehr schwer.



Beide erhalten denselben Schlüssel, denn $B^a = (g^b)^a = g^{ab} = (g^a)^b = A^b$ (alles modulo p). Eve kennt p, g, A, B – aber weder a noch b.

3.2 Sicherheit: der diskrete Logarithmus

Um s zu berechnen, müsste Eve a aus $A = g^a \bmod p$ bestimmen – den *diskreten Logarithmus*. Ein schnelles Verfahren dafür ist nicht bekannt; Eve bleibt im Wesentlichen nur Probieren:

$$g^1, g^2, g^3, \cdots \bmod 23: \quad 5, 2, 10, 4, 20, 8$$

Nach 6 Versuchen hat sie a gefunden. Bei einer Primzahl p mit über 600 Dezimalstellen ist das aussichtslos.

Man-in-the-Middle: Diffie-Hellman schützt gegen *Mitlesen*, nicht gegen *Fälschen*. Fängt Mallory A und B ab und schickt beiden eigene Werte, vereinbart er mit Alice und mit Bob je einen Schlüssel und liest alles. Abhilfe: Die öffentlichen Werte werden *signiert* (4.4) bzw. über Zertifikate geprüft.

3.3 Primitivwurzeln

Damit Eve möglichst viele Möglichkeiten durchprobieren muss, sollen die Potenzen $g^1, g^2, \ldots$ *alle* Zahlen $1, \ldots, p-1$ treffen. Die *Ordnung* von g ist der kleinste Exponent $k > 0$ mit $g^k \bmod p = 1$. Hat g die größtmögliche Ordnung $p-1$, heißt g *Primitivwurzel* (Erzeuger) modulo p. Beispiel $p = 19$:

g	g^1	g^2	g^3	g^4	g^5	g^6	g^7	g^8	g^9	g^{10}	g^{11}	g^{12}	g^{13}	g^{14}	g^{15}	g^{16}	g^{17}	g^{18}	ord
2	2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1	18
4	4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1	9
7	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	3

2 durchläuft alle 18 Reste, 4 und 7 nur einen Teil. Die Primitivwurzeln modulo 19 sind 2, 3, 10, 13, 14, 15.

Schnelltest: g ist genau dann Primitivwurzel modulo p, wenn $g^{(p-1)/q} \bmod p \neq 1$ für *jeden* Primfaktor q von $p-1$. Für $p = 19$ ist $p - 1 = 18 = 2 \cdot 3^2$, man prüft also nur g^9 und g^6:

g	$g^{18/2} = g^9$	$g^{18/3} = g^6$	Primitivwurzel?
2	18	7	ja
4	1	11	nein
7	1	1	nein
10	18	11	ja

Diffie-Hellman auf einen Blick	
Schlüssel	öffentlich p, g; geheim a (Alice), b (Bob)
Rechenvorschrift	$A = g^a$, $B = g^b$, $s = B^a = A^b \pmod p$
Sicherheit beruht auf	diskreter Logarithmus: aus $g^a \bmod p$ lässt sich a praktisch nicht berechnen
Schwachstelle	Man-in-the-Middle ohne Signatur; kleine p oder g mit kleiner Ordnung

4 RSA

4.1 Die Idee: eine Falltür

Bei Diffie-Hellman kann *niemand* die Einbahnstraße zurückgehen. Für Verschlüsselung mit öffentlichem Schlüssel braucht man eine *Falltür*: eine Einbahnstraße, die der Besitzer eines Geheimnisses doch umkehren kann. RSA (Rivest, Shamir, Adleman 1977) nutzt: Zwei große Primzahlen zu multiplizieren ist leicht, das Produkt wieder zu *zerlegen* ist sehr schwer. Wer die Zerlegung kennt, kennt die Falltür.

Die *Eulersche φ-Funktion* $\varphi(n)$ zählt die Zahlen $1, \ldots, n$, die zu n teilerfremd sind. Für $n = p \cdot q$ mit Primzahlen $p \neq q$ gilt $\varphi(n) = (p-1)(q-1)$ – das kann nur ausrechnen, wer p und q kennt.

4.2 Schlüsselerzeugung

1. Zwei Primzahlen wählen: $p = 17$, $q = 23$.
2. $n = p \cdot q = 391$ und $\varphi(n) = (p-1)(q-1) = 16 \cdot 22 = 352$.
3. Einen Exponenten e mit $\mathrm{ggT}(e, \varphi(n)) = 1$ wählen: $e = 3$.
4. $d = e^{-1} \bmod \varphi(n)$ mit dem erweiterten euklidischen Algorithmus (2.3) berechnen.

a	b	q	r	x	y
352	3	117	1	1	$0 - 117 \cdot 1 = -117$
3	1	3	0	0	1

Bézout: $1 = 352 \cdot 1 + 3 \cdot (-117)$. Modulo $\varphi(n)$ bleibt $3 \cdot (-117) \equiv 1$, also $d = -117 + 352 = 235$. Probe: $e \cdot d = 705 = 2 \cdot 352 + 1$. ✓

Öffentlicher Schlüssel $(e, n) = (3, 391)$ **Privater Schlüssel** $(d, n) = (235, 391)$. p, q und $\varphi(n)$ werden danach vernichtet: Wer sie kennt, kann d sofort ausrechnen.

4.3 Verschlüsseln und entschlüsseln

Die Nachricht ist eine Zahl m mit $0 \leq m < n$.

$$\text{verschlüsseln: } c = m^e \bmod n, \qquad \text{entschlüsseln: } m = c^d \bmod n$$

Beispiel: $m = 42$ ergibt $c = 42^3 \bmod 391 = 189$, und $189^{235} \bmod 391 = 42$ (schnelles Potenzieren!).

Dass das immer funktioniert, folgt aus dem *Satz von Euler* ($m^{\varphi(n)} \equiv 1 \pmod n$ für teilerfremde m, n): Wegen $e \cdot d = k \cdot \varphi(n) + 1$ ist $c^d = m^{ed} = (m^{\varphi(n)})^k \cdot m \equiv m$.

Quelltext 5: RSA mit kleinen Zahlen (nur zum Nachrechnen, nicht sicher!)

```
def rsa_schluessel(p, q, e):
    n = p * q
    phi = (p - 1) * (q - 1)
    d = inverse(e, phi)              # erweiterter Euklid, Abschnitt 2.3
    return (e, n), (d, n)

def rsa(x, schluessel):
    k, n = schluessel
    return modpotenz(x, k, n)        # verschlüsseln: k = e, entschlüsseln: k = d
```

4.4 Signaturen und Sicherheit

Mit RSA kann man auch *unterschreiben*. Die Idee: Nur Alice kennt d – also kann nur sie etwas mit d „verschlüsseln“. Mit ihrem öffentlichen Schlüssel e kann es dann *jeder* prüfen. Unterschrieben wird aber nicht die ganze Nachricht, sondern ihr *Hashwert*.

Was ist ein Hashwert?

Eine *Hashfunktion* macht aus einem beliebig langen Text eine Zahl fester Länge – eine Art *Fingerabdruck*.

- Gleicher Text ergibt immer denselben Hashwert.
- Schon ein geänderter Buchstabe ergibt einen völlig anderen Hashwert.
- Aus dem Hashwert lässt sich der Text nicht zurückrechnen.
- Zwei verschiedene Texte mit gleichem Hashwert findet man praktisch nicht.

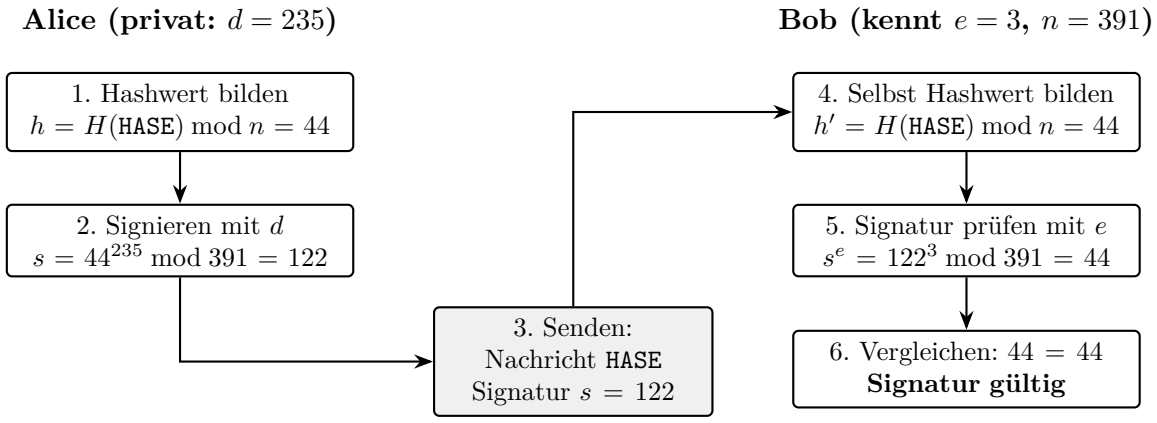
Heute verwendet man z. B. *SHA-256* (256 Bit). Das ältere MD5 gilt als gebrochen: Man kann gezielt zwei Texte mit gleichem MD5-Wert erzeugen.
Quelle: `https://de.wikipedia.org/wiki/Message-Digest_Algorithm_5`

Python stellt beides im Modul `hashlib` bereit. SHA-256 von `HASE` (hexadezimal, 64 Zeichen):

```
f85fa56cf913d60b85f6cd39d93d7ca5
53b7dce76471e2bcd8e3af96e0678f07
```

Als Dezimalzahl hat dieser Hashwert 78 Stellen (112342578645 …). Unser Spiel-RSA aus 4.2 rechnet aber nur mit Zahlen unter $n = 391$. Deshalb nehmen wir hier nur den Rest: $h = H(\texttt{HASE}) \bmod 391 = 44$. (Bei echtem RSA ist n mit 2048 Bit größer als jeder Hashwert, dort entfällt dieser Schritt.)

Ablauf einer Signatur



Fälschungsversuch: Mallory fängt die Nachricht ab und ändert sie in `HOSE`, die Signatur $s = 122$ lässt er stehen. Bob bildet $H(\texttt{HOSE})$ (beginnt mit `39c6e1c91e50432291c326f7...`, ganz anders als bei `HASE`), also $h' = 276$. Die Prüfung ergibt aber weiterhin $122^3 \bmod 391 = 44 \neq 276$: **Signatur ungültig** – Bob merkt die Änderung. Eine passende neue Signatur kann Mallory nicht berechnen, weil er d nicht kennt.

Quelltext 6: Signieren und prüfen mit Hashwert (Spielzahlen aus 4.2)

```python
import hashlib

def hashwert(text, n):
    hx = hashlib.sha256(text.encode()).hexdigest()  # oder hashlib.md5(...)
    return int(hx, 16) % n                          # nur für unser kleines n

def signieren(text, d, n):
    return modpotenz(hashwert(text, n), d, n)

def pruefen(text, s, e, n):
    return modpotenz(s, e, n) == hashwert(text, n)

s = signieren("HASE", 235, 391)       # s = 122
print(pruefen("HASE", s, 3, 391))     # True
print(pruefen("HOSE", s, 3, 391))     # False
```

	Vertraulichkeit	Integrität	Authentizität
verschlüsseln mit Bobs öffentlichem Schlüssel	ja	nein	nein
signieren mit Alices privatem Schlüssel	nein	ja	ja
beides (erst signieren, dann verschlüsseln)	ja	ja	ja

RSA auf einen Blick

Schlüssel	öffentlich (e, n), privat (d, n) mit $e \cdot d \equiv 1 \pmod{\varphi(n)}$
Rechenvorschrift	$c = m^e \bmod n$, $m = c^d \bmod n$
Sicherheit beruht auf	Faktorisierung: aus n lassen sich p und q praktisch nicht bestimmen
Schwachstelle	kleine oder bekannte Primzahlen; „Lehrbuch-RSA“ ohne Zufallsauffüllung (Padding): gleiche Nachricht ergibt gleichen Geheimtext

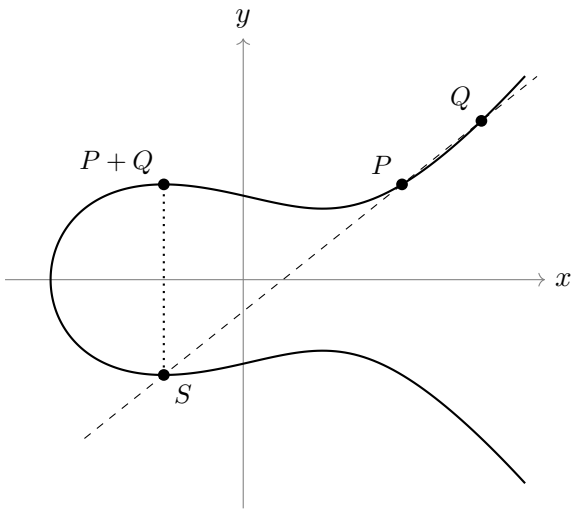
5 Elliptische Kurven

5.1 Kurven über den reellen Zahlen

Eine *elliptische Kurve* ist die Menge der Punkte $(x \mid y)$ mit

$$y^2 = x^3 + ax + b, \qquad 4a^3 + 27b^2 \neq 0$$

(die Bedingung verhindert Spitzen und Selbstüberschneidungen). Die Kurve ist symmetrisch zur x-Achse. Beispiel $y^2 = x^3 - 3x + 7$:



5.2 Punkte addieren

Auf der Kurve kann man Punkte „addieren“:

1. Gerade durch P und Q legen. Sie schneidet die Kurve in genau einem dritten Punkt S.
2. S an der x-Achse spiegeln. Das Ergebnis ist $P + Q$.

Für $P = Q$ nimmt man die *Tangente* in P (Verdoppeln, $2P$). Liegen P und Q senkrecht übereinander, gibt es keinen dritten Schnittpunkt: Das Ergebnis ist der *Punkt im Unendlichen* $\mathcal{O}$, das neutrale Element ($P + \mathcal{O} = P$). Der gespiegelte Punkt ist $-P$, es gilt $P + (-P) = \mathcal{O}$.

Rechenrezept für $P = (x_1 \mid y_1)$, $Q = (x_2 \mid y_2)$, $R = P + Q = (x_3 \mid y_3)$:

$$m = \frac{y_2 - y_1}{x_2 - x_1}\ (P \neq Q), \qquad m = \frac{3x_1^2 + a}{2y_1}\ (P = Q)$$

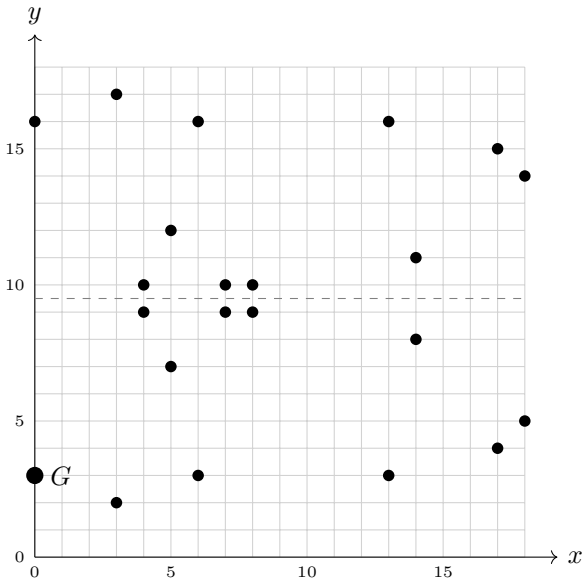
$$x_3 = m^2 - x_1 - x_2, \qquad y_3 = m\,(x_1 - x_3) - y_1 \quad \text{(Spiegelung ist schon eingerechnet)}$$

Beispiel: $P = (2 \mid 3)$, $Q = (3 \mid 5)$: $m = \frac{5-3}{3-2} = 2$, $x_3 = 4 - 2 - 3 = -1$, $y_3 = 2 \cdot (2 - (-1)) - 3 = 3$. Also $P + Q = (-1 \mid 3)$ – im Bild: Gerade $y = 2x - 1$, dritter Schnittpunkt $S = (-1 \mid -3)$, gespiegelt.

Verdoppeln von $R = (-1 \mid 3)$: $m = \frac{3 \cdot 1 - 3}{2 \cdot 3} = 0$ (waagerechte Tangente), $x_3 = 0 + 1 + 1 = 2$, $y_3 = 0 - 3 = -3$. Also $2R = (2 \mid -3)$ – das ist gerade $-P$.

5.3 Kurven über endlichen Körpern $\mathbb{F}_p$

Über den reellen Zahlen entstehen schnell Brüche mit riesigen Nennern. Die Kryptographie rechnet deshalb *modulo einer Primzahl* p: Es gelten dieselben Formeln, und „durch z teilen“ heißt „mit $z^{-1} \bmod p$ malnehmen“ (2.3). Die Kurve $y^2 = x^3 + 2x + 9$ über $\mathbb{F}_{19}$ besteht aus 22 Punkten plus $\mathcal{O}$:



Aus der Kurve ist eine symmetrische Punktwolke geworden (Spiegelachse $y = 9{,}5$, denn $-y \equiv 19 - y$).

Vorgerechnet: $G = (0\,|\,3)$ verdoppeln.

$$m = \frac{3 \cdot 0^2 + 2}{2 \cdot 3} = \frac{2}{6}, \qquad 6^{-1} \bmod 19 = 16 \text{ (denn } 6 \cdot 16 = 96 = 5 \cdot 19 + 1), \qquad m = 2 \cdot 16 \bmod 19 = 13$$

$$x_3 = 13^2 - 0 - 0 = 169 \bmod 19 = 17, \qquad y_3 = 13 \cdot (0 - 17) - 3 = -224 \bmod 19 = 4$$

Also $2G = (17\,|\,4)$. So weiter addiert erhält man alle Vielfachen $k \cdot G$:

k	1	2	3	4	5	6	7	8
$k \cdot G$	$(0\,\|\,3)$	$(17\,\|\,4)$	$(7\,\|\,10)$	$(13\,\|\,3)$	$(6\,\|\,16)$	$(14\,\|\,11)$	$(3\,\|\,17)$	$(4\,\|\,10)$

k	9	10	11	12	13	14	15	16
$k \cdot G$	$(5\,\|\,12)$	$(18\,\|\,14)$	$(8\,\|\,9)$	$(8\,\|\,10)$	$(18\,\|\,5)$	$(5\,\|\,7)$	$(4\,\|\,9)$	$(3\,\|\,2)$

k	17	18	19	20	21	22	23
$k \cdot G$	$(14\,\|\,8)$	$(6\,\|\,3)$	$(13\,\|\,16)$	$(7\,\|\,9)$	$(17\,\|\,15)$	$(0\,\|\,16)$	$\mathcal{O}$

Die Gruppe hat 23 Elemente, $23 \cdot G = \mathcal{O}$. Weil 23 eine Primzahl ist, erzeugt *jeder* Punkt außer $\mathcal{O}$ die ganze Gruppe – ein guter Generator für die Kryptographie.

Vielfache schnell berechnen (double-and-add): $k \cdot G$ berechnet man wie das schnelle Potenzieren (2.4), nur mit „verdoppeln“ statt „quadrieren“ und „addieren“ statt „multiplizieren“. Die Umkehrung – aus $k \cdot G$ die Zahl k finden – heißt *diskreter Logarithmus auf der Kurve (ECDLP)* und ist noch schwerer als bei Diffie-Hellman.

Quelltext 7: Punktaddition über $\mathbb{F}_p$ (None steht für $\mathcal{O}$)

```
def addiere(P, Q, a, p):
    if P is None:
        return Q
    if Q is None:
        return P
    x1, y1 = P
    x2, y2 = Q
    if x1 == x2 and (y1 + y2) % p == 0:
        return None                                  # P + (-P) = O
    if P == Q:
        m = (3 * x1 * x1 + a) * inverse(2 * y1, p) % p   # Tangente
    else:
        m = (y2 - y1) * inverse(x2 - x1, p) % p          # Sekante
    x3 = (m * m - x1 - x2) % p
    y3 = (m * (x1 - x3) - y1) % p
    return (x3, y3)
```

5.4 Verschlüsseln mit elliptischen Kurven (EC-ElGamal)

Öffentlich sind Kurve und Generator G. Die Nachricht wird vorher als Kurvenpunkt X codiert (z. B. über ein Codebuch: jedem Buchstaben ein Punkt).

1. **Alice** wählt geheim k und veröffentlicht $A = k \cdot G$.
2. **Bob** wählt eine Einmalzahl m und schickt $T = (B, Y)$ mit $B = m \cdot G$ und $Y = X + m \cdot A$.
3. **Alice** rechnet $C = k \cdot B$ $(= k \cdot m \cdot G = m \cdot A)$ und erhält $X = Y - C = Y + (-C)$.

Beispiel auf der Kurve aus 5.3: $k = 7$, also $A = 7 \cdot G = (3 \,|\, 17)$. Bob will $X = (18 \,|\, 14)$ senden und wählt $m = 5$: $B = 5 \cdot G = (6 \,|\, 16)$, $m \cdot A = (8 \,|\, 10)$, $Y = X + m \cdot A = (0 \,|\, 16)$. Er schickt $T = ((6 \,|\, 16), (0 \,|\, 16))$. Alice: $C = 7 \cdot B = (8 \,|\, 10)$, $-C = (8 \,|\, 9)$, $Y + (-C) = (18 \,|\, 14) = X$. ✓

Sicherheitsniveau (symmetrischer Schlüssel)	128 Bit	192 Bit	256 Bit
RSA, Diffie-Hellman modulo p	3072 Bit	7680 Bit	15360 Bit
elliptische Kurven	256 Bit	384 Bit	512 Bit

Vergleichbare Schlüssellängen nach NIST SP 800-57 Teil 1, Rev. 5, Tabelle 2. Quelle:
`https://csrc.nist.gov/pubs/sp/800/57/pt1/r5/final`

Für dieselbe Sicherheit genügen bei elliptischen Kurven viel kürzere Schlüssel: Für das ECDLP kennt man keine Abkürzung wie für Faktorisierung und diskreten Logarithmus modulo p. Kürzere Schlüssel bedeuten schnellere Rechnungen und weniger Speicher – deshalb nutzen Smartphones, Ausweise und moderne Webverbindungen elliptische Kurven.

EC-ElGamal auf einen Blick	
Schlüssel	öffentlich Kurve, G, $A = k \cdot G$; geheim k
Rechenvorschrift	$T = (m \cdot G,\ X + m \cdot A)$, $X = Y - k \cdot B$
Sicherheit beruht auf	ECDLP: aus $k \cdot G$ lässt sich k praktisch nicht bestimmen
Schwachstelle	dieselbe Einmalzahl m zweimal benutzt: Aus zwei Geheimtexten lässt sich die Differenz der Klartexte ablesen

6 Überblick

Verfahren	Art	Sicherheit beruht auf	Angriff / Schwäche
Caesar, multiplikativ	symmetrisch	–	alle Schlüssel probieren, Häufigkeitsanalyse
Vigenère	symmetrisch	Länge des Schlüsselworts	Kasiski, Friedman, dann Häufigkeitsanalyse
One-Time-Pad	symmetrisch	echtem Zufall, einmaliger Gebrauch	keiner – aber Schlüsselübergabe
Diffie-Hellman	Schlüssel-tausch	diskretem Logarithmus mod p	Man-in-the-Middle ohne Signatur
RSA	asymmetrisch	Schwierigkeit der Faktorisierung	kleine p, q; Lehrbuch-RSA ohne Padding
EC-ElGamal	asymmetrisch	diskretem Logarithmus auf der Kurve (ECDLP)	Einmalzahl mehrfach benutzt

Werkzeuge, die immer wieder vorkommen

- **Rechnen modulo** n: überall – Zahlen bleiben klein, „Ziffernblatt“ statt Zahlengerade.
- **Erweiterter Euklid**: Inverses für die multiplikative Chiffre, d bei RSA, Division auf der Kurve.
- **Schnelles Potenzieren / double-and-add**: Diffie-Hellman, RSA, $k \cdot G$ auf der Kurve.

Selbstkontrolle

1. Warum hat die multiplikative Chiffre modulo 26 nur 12 brauchbare Schlüssel?
2. Warum verrät der Abstand zweier gleicher Geheimtextfolgen etwas über die Schlüssellänge bei Vigenère?
3. Welche Werte kennt Eve beim Diffie-Hellman-Tausch, und warum hilft ihr das nicht?
4. Warum muss man bei RSA p und q geheim halten, obwohl nur n veröffentlicht wird?
5. Was bedeutet „teilen“ beim Rechnen auf einer Kurve über $\mathbb{F}_p$?

Lösungshinweise: 1. Nur m mit $\mathrm{ggT}(m, 26) = 1$ haben ein Inverses. 2. Gleicher Klartext trifft nur dann gleich verschlüsselt, wenn der Abstand ein Vielfaches der Schlüssellänge ist. 3. p, g, A, B; für s bräuchte sie a oder b (diskreter Logarithmus). 4. Aus p, q folgt $\varphi(n)$ und damit sofort d. 5. Mit dem Inversen modulo p multiplizieren.

7 Übungen

Aufgabe 1 (multiplikative Chiffre). Der Geheimtext `FHOUZRAWFU` wurde mit $m = 5$ verschlüsselt. Bestimmen Sie $5^{-1} \bmod 26$ und entschlüsseln Sie.

Aufgabe 2 (Vigenère). Verschlüsseln Sie `WINTERFERIEN` mit dem Schlüsselwort `EIS`.

Aufgabe 3 (Kasiski). In einem Vigenère-Geheimtext treten wiederholte Folgen mit den Abständen 42, 63, 105, 84 auf. Welche Schlüssellängen kommen in Frage, welche ist am wahrscheinlichsten?

Aufgabe 4 (erweiterter Euklid). Berechnen Sie $17^{-1} \bmod 120$ mit dem Rechenschema aus 2.3 und machen Sie die Probe.

Aufgabe 5 (schnelles Potenzieren). Berechnen Sie $11^{23} \bmod 47$ mit einer Tabelle wie in 2.4.

Aufgabe 6 (Diffie-Hellman). Öffentlich sind $p = 29$ und $g = 2$. Alice wählt $a = 9$, Bob $b = 13$. Berechnen Sie A, B und den gemeinsamen Schlüssel s auf beiden Wegen.

Aufgabe 7 (Primitivwurzeln). Prüfen Sie mit dem Schnelltest, welche der Zahlen 2, 4, 3 Primitivwurzeln modulo 29 sind.

Aufgabe 8 (RSA). Erzeugen Sie aus $p = 19$, $q = 29$ und $e = 5$ ein Schlüsselpaar. Verschlüsseln Sie $m = 100$ und geben Sie an, wie man wieder entschlüsselt.

Aufgabe 9 (elliptische Kurve). Gegeben ist die Kurve $y^2 = x^3 + 2x + 9$ über $\mathbb{F}_{19}$ aus 5.3.

(a) Zeigen Sie, dass $Q = (3 \mid 17)$ auf der Kurve liegt, und berechnen Sie $2Q$.
(b) Alice hat den geheimen Schlüssel $k = 4$. Sie erhält $T = ((7 \mid 10),\ (13 \mid 3))$. Entschlüsseln Sie den Klartextpunkt X.

8 Lösungen

Lösung 1. $5 \cdot 21 = 105 = 4 \cdot 26 + 1$, also $5^{-1} = 21$. Entschlüsseln: $x = 21 \cdot y \bmod 26$.

Geheimtext	F	H	O	U	Z	R	A	W	F	U
y	5	7	14	20	25	17	0	22	5	20
$21y \bmod 26$	1	17	8	4	5	19	0	20	1	4
Klartext	B	R	I	E	F	T	A	U	B	E

Klartext: BRIEFTAUBE.

Lösung 2. E = 4, I = 8, S = 18:

Klartext	W	I	N	T	E	R	F	E	R	I	E	N
Schlüssel	E	I	S	E	I	S	E	I	S	E	I	S
Geheimtext	A	Q	F	X	M	J	J	M	J	M	M	F

Geheimtext: AQFXMJJMJMMF.

Lösung 3. $42 = 2 \cdot 3 \cdot 7$; $63 = 3 \cdot 3 \cdot 7$; $105 = 3 \cdot 5 \cdot 7$; $84 = 2 \cdot 2 \cdot 3 \cdot 7$. Gemeinsame Teiler aller Abstände: 1, 3, 7, 21. Schlüssellänge 1 wäre eine Caesar-Chiffre (Häufigkeitsanalyse würde sofort helfen). Am wahrscheinlichsten ist der ggT 21; möglich sind auch 7 oder 3 – das prüft man, indem man die Teiltexte untersucht.

Lösung 4.

a	b	q	r	x	y
120	17	7	1	1	$0 - 7 \cdot 1 = -7$
17	1	17	0	0	1

Bézout: $1 = 120 \cdot 1 + 17 \cdot (-7)$, modulo 120 also $17 \cdot (-7) \equiv 1$ und $-7 + 120 = 113$, also $17^{-1} \bmod 120 = 113$. Probe: $17 \cdot 113 = 1921 = 16 \cdot 120 + 1$. ✓

Lösung 5.

ba	ex	pot	ex ungerade?
11	23	1	ja
11	22	11	nein
27	11	11	ja
27	10	15	nein
24	5	15	ja
24	4	31	nein
12	2	31	nein
3	1	31	ja
3	0	**46**	–

$11^{23} \bmod 47 = 46$.

Lösung 6. $A = 2^9 \bmod 29 = 19$, $B = 2^{13} \bmod 29 = 14$. $s = B^a = 14^9 \bmod 29 = 3$ und $s = A^b = 19^{13} \bmod 29 = 3$.

Lösung 7. $p - 1 = 28 = 2^2 \cdot 7$, zu prüfen sind $g^{28/2} = g^{14}$ und $g^{28/7} = g^4$:

g	$g^{14} \bmod 29$	$g^4 \bmod 29$	Primitivwurzel?
2	28	16	ja
4	1	24	nein (Ordnung 14)
3	28	23	ja

$4 = 2^2$ kann keine Primitivwurzel sein: $4^{14} = 2^{28} = 1$.

Lösung 8. $n = 19 \cdot 29 = 551$, $\varphi(n) = 18 \cdot 28 = 504$, $\mathrm{ggT}(5, 504) = 1$.

a	b	q	r	x	y
504	5	100	4	-1	$1 - 100 \cdot (-1) = 101$
5	4	1	1	1	$0 - 1 \cdot 1 = -1$
4	1	4	0	0	1

$d = 101$ (Probe $5 \cdot 101 = 505 = 504 + 1$). Öffentlich $(5, 551)$, privat $(101, 551)$. $c = 100^5 \bmod 551 = 180$; entschlüsseln: $180^{101} \bmod 551 = 100$.

Lösung 9. (a) $y^2 = 17^2 = 289 \bmod 19 = 4$ und $x^3 + 2x + 9 = 27 + 6 + 9 = 42 \bmod 19 = 4$. ✓
$m = \frac{3 \cdot 3^2 + 2}{2 \cdot 17} = \frac{29}{34} \equiv \frac{10}{15}$; $15^{-1} \bmod 19 = 14$ (denn $15 \cdot 14 = 210 = 11 \cdot 19 + 1$); $m = 10 \cdot 14 \bmod 19 = 7$.
$x_3 = 49 - 3 - 3 = 43 \bmod 19 = 5$, $y_3 = 7 \cdot (3 - 5) - 17 = -31 \bmod 19 = 7$. Also $2Q = (5 \,|\, 7)$.

(b) $C = k \cdot B = 4 \cdot (7 \,|\, 10) = (8 \,|\, 10)$ (z. B. zweimal verdoppeln), $-C = (8 \,|\, 9)$, $X = Y + (-C) = (13 \,|\, 3) + (8 \,|\, 9) = (4 \,|\, 9)$.