Symmetrische Verschlüsselung

Bob möchte eine verschlüsselte Nachricht von Alice empfangen.
Dazu machen sie sich ein geheimes Passwort aus („Privater Schlüssel“).

1) Alice verschlüsselt ihre Nachricht mit diesem Schlüssel.

2) Alice schickt die verschlüsselte Nachricht an Bob.

3) Bob entschlüsselt die empfangene Nachricht mit diesem Schlüssel.

Das ist das Grundprinzip von **symmetrischer Verschlüsselung**. Sie bringt Probleme mit sich:

⚠ Jede Person, die den Schlüssel kennt, kann die Nachricht auch entschlüsseln.
Alice und Bob müssen also *beide* sorgsam mit diesem privaten Schlüssel umgehen.

⚠ Wie vereinbaren Alice und Bob den privaten Schlüssel? Alice wohnt in Australien. Bob wohnt in Brasilien.

Caesar-Verschlüsselung

Ein historisches Beispiel für symmetrische Verschlüsselung ist die Caesar-Verschlüsselung.
Dabei wird jeder Buchstabe in der Nachricht nach folgendem Muster ersetzt:

Klartext:	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
Geheimtext:	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

G A L L I E N
J D O O L H Q

Der private Schlüssel bei diesem Verfahren ist eine natürliche Zahl von 1 bis 26.
Diese Zahl gibt an, um wie viele Stellen das Alphabet verschoben wird.

Gaius Julius Caesar soll – wie im Beispiel oben – für militärische Nachrichten den Schlüssel 3 verwendet haben.

Zum Entschlüsseln verschiebt man das Alphabet um den privaten Schlüssel in die andere Richtung:

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

J D O O L H Q
G A L L I E N

Welche Schwachstelle hat diese Verschlüsselung, auch wenn Alice und Bob den Schlüssel geheim halten?
Es gibt nur 26 verschiedene Schlüssel.
Man kann also den richtigen Schlüssel durch Probieren relativ schnell finden. („Brute Force“)

Monoalphabetische Verschlüsselung

Bei monoalphabetischen Verschlüsselungen werden die Buchstaben im Alphabet beliebig vertauscht.
Ein privater Schlüssel zum Verschlüsseln und Entschlüsseln von Nachrichten kann dann so aussehen:

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

G A L L I E N
T L I I J Z P

Wie viele solche Schlüssel ermöglichen die 26 Buchstaben im deutschen Alphabet?

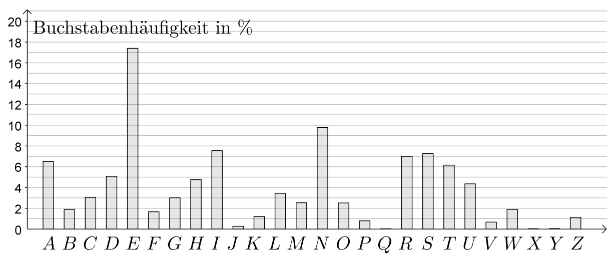
$26! = 26 \cdot 25 \cdot 24 \cdot \ldots \cdot 2 \cdot 1 \approx 4 \cdot 10^{26}$

Mehr dazu findest du am Arbeitsblatt – Kombinatorik.

Trotz der großen Schlüsselanzahl hat jede monoalphabetische Verschlüsselung eine Schwachstelle.
Rechts siehst du die prozentuellen Häufigkeiten der Buchstaben in typischen deutschsprachigen Texten:

Du fängst eine lange verschlüsselte Nachricht ab.
Wie würdest du versuchen, den Code zu knacken?

Zum Beispiel kann man die Häufigkeiten der Buchstaben im Geheimtext mit den Buchstabenhäufigkeiten vergleichen, um Rückschlüsse zu ziehen.



Hier kannst du deine Fertigkeiten beim Codeknacken auf die Probe stellen.

Asymmetrische Verschlüsselung

Bob möchte eine verschlüsselte Nachricht von Alice empfangen.
Dazu erstellt Bob vorher zwei verschiedene Schlüssel:

- **Öffentlicher Schlüssel:** Dieser Schlüssel ist *nicht* geheim. „Public Key“
  Jede Person kann ihn verwenden, um an Bob verschlüsselte Nachrichten zu senden.
  Verschlüsselte Nachrichten können mit dem öffentlichen Schlüssel *nicht* entschlüsselt werden.
- **Privater Schlüssel:** Diesen Schlüssel muss *nur* Bob geheim halten. „Private Key“
  Bob kann mit *seinem* privaten Schlüssel jede Nachricht entschlüsseln,
  die mit *seinem* öffentlichen Schlüssel verschlüsselt wurde.

Das ist das Grundprinzip von **asymmetrischer Verschlüsselung**. „Public-Key-Verschlüsselung“
Asymmetrische Verschlüsselung hat Vorteile gegenüber symmetrischer Verschlüsselung:

1) Nur der Empfänger Bob muss *seinen* privaten Schlüssel geheim halten.

2) Es muss *kein* geheimer Schlüssel ausgetauscht werden, um verschlüsselte Nachrichten zu versenden.

In der Praxis sind symmetrische Verfahren schneller als asymmetrische Verfahren. Deshalb wird hybride Verschlüsselung verwendet:
Zuerst tauschen Alice und Bob einen privaten Schlüssel mit *asymmetrischer* Verschlüsselung aus.
Danach verwenden sie diesen privaten Schlüssel, um Nachrichten mit *symmetrischer* Verschlüsselung zu versenden.

RSA-Verfahren – Schlüsselpaar erzeugen

Das **RSA-Verfahren** ist ein asymmetrisches Verschlüsselungsverfahren.
Der Algorithmus wurde im Jahr 1977 von **R**ivest, **S**hamir und **A**dleman veröffentlicht.

Einen öffentlichen Schlüssel und den zugehörigen privaten Schlüssel kannst du so berechnen:

1) Wähle (geheim) zwei verschiedene Primzahlen p und q.

2) Berechne $n = p \cdot q$.

3) Berechne $\varphi(n) = (p-1)\cdot(q-1)$. Mehr zur Eulerschen φ-Funktion findest du am AB – Kleiner Satz von Fermat.

4) Wähle eine Zahl $e > 1$, die zu $\varphi(n)$ teilerfremd ist, also eine Zahl e mit $\operatorname{ggT}(e, \varphi(n)) = 1$.
Berechne mit dem Euklidischen Algorithmus eine ganze Zahl $d > 1$ mit $e \cdot d \equiv 1 \mod \varphi(n)$.

Die beiden Zahlen $\boldsymbol{n}$ und $\boldsymbol{e}$ sind der **öffentliche Schlüssel**. Die Zahl $\boldsymbol{d}$ ist der **private Schlüssel**.

RSA-Verfahren – Schlüsselpaar erzeugen

Erzeuge den öffentlichen und den privaten RSA-Schlüssel mit $p = 7$, $q = 13$ und $e = 11$.

$n = 7 \cdot 13 = 91 \qquad \varphi(n) = 6 \cdot 12 = 72$

$$1 = 6 - 5 \cdot 1 = 6 - (11 - 6 \cdot 1) \cdot 1 =$$
$$= 6 \cdot 2 - 11 \cdot 1 = (72 - 11 \cdot 6) \cdot 2 - 11 \cdot 1 =$$
$$= 72 \cdot 2 - 11 \cdot 13$$

$$\Longrightarrow 1 \equiv -11 \cdot 13 \mod 72 \Longrightarrow d = -13 \equiv 59 \mod 72$$

Euklidischer Algorithmus:

$$72 = 11 \cdot \boxed{6} + \boxed{6}$$
$$11 = \boxed{6} \cdot \boxed{1} + \boxed{5}$$
$$\boxed{6} = \boxed{5} \cdot \boxed{1} + \boxed{1}$$
$$\boxed{5} = \boxed{1} \cdot \boxed{5} + \boxed{0}$$

Öffentlicher Schlüssel: $n = 91$, $e = 11$ Privater Schlüssel: $d = 59$

RSA-Verfahren – Nachrichten verschlüsseln

Alice möchte an Bob eine verschlüsselte Nachricht senden.

1) Alice erhält den öffentlichen Schlüssel (n, e) von Bob.

2) Dann kann Alice damit jede natürliche Zahl m mit $1 \leq m < n$ an Bob übertragen.
Dafür berechnet Alice $m^e \mod n$ und sendet das Ergebnis c mit $1 \leq c < n$ an Bob.

Jede Nachricht ist im Computer als Binärzahl – zum Beispiel 100101110101 – gespeichert. Text versenden ist wie Zahlen versenden. Wenn die Nachricht zu lange ist, kann sie vorher in Teile zerlegt werden. In der Praxis werden jeder Nachricht vor der Verschlüsselung noch zufällige Teile hinzugefügt, damit *gleiche unverschlüsselte* Nachrichten *verschiedene verschlüsselte* Nachrichten ergeben.

RSA-Verfahren – Nachrichten verschlüsseln

Bob hat den öffentlichen Schlüssel $n = 91$, $e = 11$. Verschlüssele damit die Nachricht $m = 42$.
Um $42^{11} \mod 91$ zu berechnen, zerlegen wir den Exponenten in Zweier-Potenzen: $11 = 2^3 + 2^1 + 2^0$

1) $42^2 = 19 \cdot 91 + 35 \equiv 35 \mod 91$

2) $42^4 = (42^2)^2 \equiv 35^2 = 13 \cdot 91 + 42 \equiv 42 \mod 91$

3) $42^8 = (42^4)^2 \equiv 42^2 \equiv 35 \mod 91$

$$\implies 42^{11} = 42^8 \cdot 42^2 \cdot 42^1 \equiv 35 \cdot 35 \cdot 42 = 565 \cdot 91 + 35 \equiv 35 \mod 91 \implies c = 35$$

RSA-Verfahren – Nachrichten entschlüsseln

Bob möchte die verschlüsselte Nachricht c entschlüsseln.

1) Bob berechnet $c^d \mod n$ mit seinem privaten Schlüssel d.

2) Das Ergebnis ist die unverschlüsselte Nachricht m.

Der Beweis dafür ist auf der nächsten Seite.

RSA-Verfahren – Nachrichten entschlüsseln

Bob hat den öffentlichen Schlüssel $n = 91$, $e = 11$ und den privaten Schlüssel $d = 59$.
Zerlege 59 in Zweier-Potenzen, und entschlüssele damit für Bob die Nachricht $c = 35$.

1) $35^2 = 91 \cdot 13 + 42 \equiv 42 \mod 91$

2) $35^4 = (35^2)^2 \equiv 42^2 = 91 \cdot 19 + 35 \equiv 35 \mod 91$

3) $35^8 = (35^4)^2 \equiv 35^2 \equiv 42 \mod 91$

4) $35^{16} = (35^8)^2 \equiv 42^2 \equiv 35 \mod 91$

5) $35^{32} = (35^{16})^2 \equiv 35^2 \equiv 42 \mod 91$

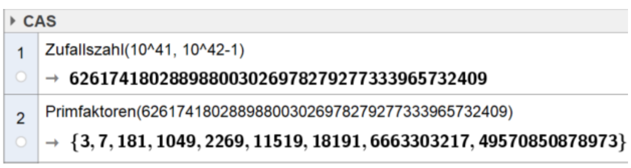
$$\implies 35^{59} = 35^{32} \cdot 35^{16} \cdot 35^8 \cdot 35^2 \cdot 35^1 \equiv 42^3 \cdot 35^2 \equiv 42^4 \equiv 42 \mod 91 \implies m = 42$$

Was sind *große* Primzahlen?

Die Sicherheit des RSA-Verfahrens beruht darauf, dass wir mit unseren aktuellen technischen Mitteln aus einer großen Zahl $n = p \cdot q$ die beiden Primfaktoren p und q *nicht effizient* berechnen können.
Sonst könnte jede Person aus dem öffentlichen Schlüssel auch den privaten Schlüssel berechnen und die Nachrichten entschlüsseln.

42-stellige Zahlen zerlegt GeoGebra im Bruchteil einer Sekunde in ihre Primfaktoren:

626 174 180 288 988 003 026 978 279 277 333 965 732 409 =
3·7·181·1049·2269·11 519·18 191·6 663 303 217·49 570 850 878 973



In der Praxis werden deshalb beim RSA-Verfahren Primzahlen mit rund 300 Stellen verwendet.

Primfaktorzerlegung

Es sind p und q *verschiedene* Primzahlen.
Erkläre, warum dann die folgende Äquivalenz für alle ganzen Zahlen a und b gilt:

$$a \equiv b \mod p \cdot q \iff a \equiv b \mod p \quad \text{und} \quad a \equiv b \mod q$$

$a \equiv b \mod p \cdot q \iff p \cdot q \mid a - b$
Links und rechts steht: „Die Primfaktorzerlegung von $a - b$ enthält beide Primzahlen p und q."

RSA-Verfahren (Beweis)

Warum funktioniert das **RSA-Verfahren**?
Dafür müssen wir für alle Zahlen m mit $1 \leq m < n$ die Kongruenz

$$\boldsymbol{(m^e)^d \equiv m \mod n} \tag{1}$$

zeigen, wobei der private Schlüssel d so gewählt wurde, dass $e \cdot d \equiv 1 \mod \varphi(n)$ gilt.
Es gibt also eine ganze Zahl k mit $e \cdot d = 1 + k \cdot \varphi(n)$. Statt (1) können wir somit auch zeigen:

$$m \cdot m^{k \cdot \varphi(n)} \equiv m \mod n \tag{2}$$

Fall 1: m und n sind teilerfremde Zahlen.
Erkläre, warum dann (2) aus dem Satz von Euler, also $m^{\varphi(n)} \equiv 1 \mod n$, folgt:

$$m \cdot m^{k \cdot \varphi(n)} = m \cdot \left(m^{\varphi(n)}\right)^k \equiv m \cdot 1^k = m \mod n \checkmark$$

Fall 2: m und $n = p \cdot q$ sind *nicht* teilerfremd.
Da p und q verschiedene Primzahlen sind, können wir statt (2) auch zeigen, dass gilt:

$$m \cdot m^{k \cdot \varphi(n)} \equiv m \mod p \qquad \text{und} \qquad m \cdot m^{k \cdot \varphi(n)} \equiv m \mod q \tag{3}$$

Die *kleinste* positive natürliche Zahl s, für die $\text{ggT}(s, p) > 1$ und $\text{ggT}(s, q) > 1$ gilt, ist $s = p \cdot q$.
Da $m < p \cdot q$ und $\text{ggT}(m, p \cdot q) > 1$ gilt, bleiben also nur 2 Möglichkeiten:

i) $\text{ggT}(m, q) = 1$ und $p \mid m$ oder **ii)** $\text{ggT}(m, p) = 1$ und $q \mid m$

Wir zeigen jetzt (3) im Fall **i)**. Im Fall **ii)** funktioniert es genauso. p und q vertauschen nur ihre Rollen.

$$p \mid m \implies \underbrace{m \cdot m^{k \cdot \varphi(n)}}_{\equiv 0} \equiv \underbrace{m}_{\equiv 0} \mod p \checkmark$$

Auf dem Arbeitsblatt – Kleiner Satz von Fermat haben wir erklärt, warum $\varphi(p \cdot q) = \varphi(p) \cdot \varphi(q)$ gilt.
Wegen $\text{ggT}(m, q) = 1$ können wir den Satz von Euler verwenden: $m^{\varphi(q)} \equiv 1 \mod q$
Erkläre damit, warum $m \cdot m^{k \cdot \varphi(n)} \equiv m \mod q$ gilt:

$$m \cdot m^{k \cdot \varphi(n)} = m \cdot \left(m^{\varphi(q)}\right)^{k \cdot \varphi(p)} \equiv m \cdot 1^{k \cdot \varphi(p)} = m \mod q \checkmark$$

RSA-Signatur

Das RSA-Verfahren kann wegen $(m^d)^e = (m^e)^d \equiv m \mod n$ auch umgekehrt verwendet werden:

1) Bob verschlüsselt seine Nachricht mit seinem privaten Schlüssel d. Er *unterschreibt* damit die Nachricht.

2) Alice prüft mit dem öffentlichen Schlüssel e von Bob, ob die Nachricht tatsächlich von ihm kommt.
Wenn ja, dann ist $(m^d)^e \equiv m \mod n$. Wenn nein, dann sollte die entschlüsselte Nachricht $(m^d)^e$ keinen Sinn ergeben.