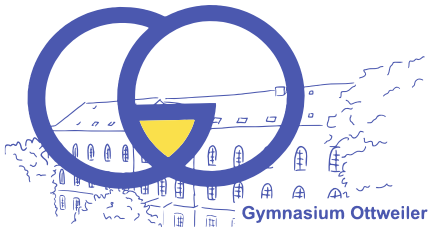
Kapitel 8

Kryptographie

Skript Informatik · Grundkurs



Wenn Sie eine Nachricht verschicken, nimmt sie einen Weg, den Sie nicht kontrollieren: durch ein WLAN, über ein Dutzend Router, durch Leitungen, die weder Ihnen noch dem Empfänger gehören. An jeder dieser Stationen kann jemand mitlesen. Das lässt sich nicht verhindern. Verhindern lässt sich nur, dass der Mitleser etwas versteht.

Darum geht es in der Kryptographie. Sie ist längst keine Sache von Militär und Geheimdiensten mehr, sondern läuft ständig unbemerkt mit: beim Aufruf einer Webseite, beim Bezahlen, beim Entsperren des Handys, bei jeder Messenger-Nachricht.

Dieses Kapitel geht chronologisch vor, und das hat einen Grund: Jedes Verfahren ist die Antwort auf die Schwäche des vorigen, und jeder Angriff nutzt eine Eigenschaft aus, die die Erfinder für harmlos hielten. Wer diese Kette einmal verfolgt hat, versteht auch, warum heutige Verfahren so aussehen, wie sie aussehen.

Lernziele

Am Ende dieses Kapitels können Sie

- die Schutzziele Vertraulichkeit, Integrität und Authentizität unterscheiden und Angriffe darauf einordnen,
- klassische Verfahren (Transposition, mono- und polyalphabetische Substitution) erklären, anwenden und programmieren,
- eine Häufigkeitsanalyse und den Kasiski-Test durchführen und begründen, warum sie funktionieren,
- erklären, warum das One-Time-Pad sicher, aber unpraktisch ist,
- den Unterschied zwischen symmetrischen und asymmetrischen Verfahren erläutern,
- den Schlüsseltausch von Diffie und Hellman an kleinen Zahlen durchführen und seine Grundidee erklären,
- den Man-in-the-Middle-Angriff beschreiben und begründen, warum Mathematik allein ihn nicht verhindert,
- das RSA-Verfahren an kleinen Zahlen vollständig durchrechnen,
- beschreiben, wie Signatur, Hashfunktion und Hybridverfahren beim Aufruf einer Webseite zusammenwirken.

8.1 Schutzziele und Angriffe

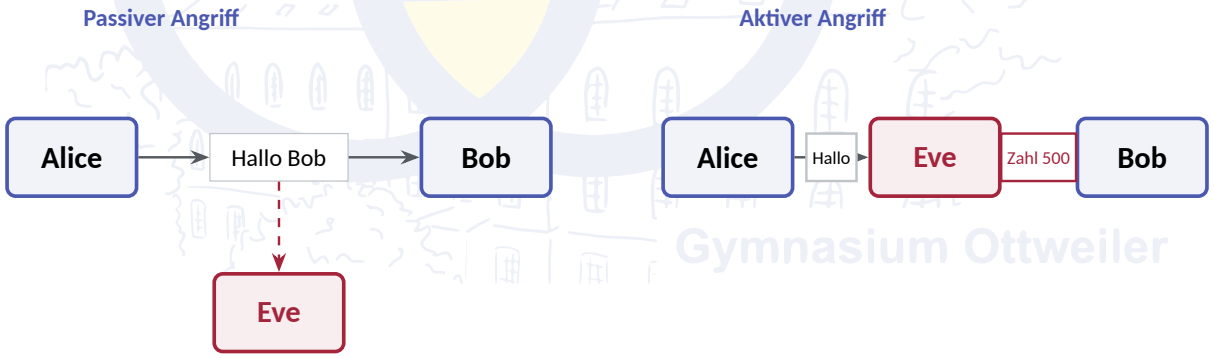
Bevor man ein Verfahren auswählt, muss klar sein, wovor es eigentlich schützen soll. Die IT-Sicherheit unterscheidet drei Grundforderungen.

Schutzziel	Leitfrage	Beispiel für eine Verletzung
Vertraulichkeit	Kann ein Dritter die Nachricht lesen?	Jemand liest im offenen WLAN die Bestellung mit.
Integrität	Ist die Nachricht unverändert angekommen?	Die Kontonummer in einer Überweisung wird unterwegs ersetzt.
Authentizität	Stammt die Nachricht wirklich vom angegebenen Absender?	Eine Mail gibt sich als Nachricht der Schulleitung aus.

Häufig kommt als viertes Ziel die **Verbindlichkeit** hinzu: Der Absender soll später nicht abstreiten können, die Nachricht geschickt zu haben.

Angriffe auf diese Ziele teilt man in zwei Gruppen:

- **Passiver Angriff.** Der Angreifer liest nur mit. Er verletzt die Vertraulichkeit, verändert aber nichts. Solche Angriffe hinterlassen keine Spuren und sind darum kaum zu entdecken – man kann sie nur *verhindern*, indem man den Inhalt unlesbar macht.
- **Aktiver Angriff.** Der Angreifer verändert, wiederholt, verzögert oder unterdrückt Nachrichten oder gibt sich als jemand anderes aus (Maskerade). Solche Angriffe lassen sich nicht verhindern, denn die Leitung gehört dem Angreifer genauso wie uns. Man kann sie nur *erkennbar* machen.



Eve liest mit und ändert nichts. Keine Spur — nur zu *verhindern*, nicht zu erkennen.

Eve ersetzt, wiederholt oder unterdrückt. Nicht zu verhindern — nur *erkennbar* zu machen.

Passiver und aktiver Angriff. Links stiehlt Eve nur Information, rechts greift sie in den Verkehr ein.

Merke

Passive Angriffe muss man verhindern, aktive Angriffe muss man erkennen. Verschlüsselung allein schützt nur die Vertraulichkeit – für Integrität und Authentizität braucht es zusätzliche Werkzeuge (Abschnitt 8.11).

Arbeitsauftrag 1

1. Ordnen Sie zu, welches Schutzziel jeweils verletzt wird: (a) Ein Angreifer schneidet den Netzwerkverkehr mit. (b) Eine Bestellung wird von 1 auf 100 Stück geändert. (c) Ein Erpresser verschlüsselt alle Dateien einer Firma. (d) Eine gefälschte Absenderadresse verschickt Rechnungen.
2. Begründen Sie, warum ein aktiver Angriff meist einen passiven voraussetzt.
3. Nennen Sie für jedes der drei Schutzziele eine Situation aus Ihrem eigenen Alltag, in der Sie sich darauf verlassen.

8.2 Grundbegriffe

Die Wörter stammen aus dem Griechischen: *kryptós* (verborgen), *gráphein* (schreiben), *lógos* (Wort, Lehre).

- **Kryptographie** ist die Lehre vom Entwurf von Verschlüsselungsverfahren.
- **Kryptoanalyse** ist die Lehre vom Brechen dieser Verfahren.
- **Kryptologie** ist der Oberbegriff für beides. Wer gute Verfahren bauen will, muss angreifen können – beide Seiten gehören untrennbar zusammen.

Der **Klartext** ist die lesbare Nachricht, der **Geheimtext** (auch: Chiffrat) ihre verschlüsselte Form. Der Vorgang heißt **Chiffrierung** bzw. **Dechiffrierung**, das zugrunde liegende Verfahren **Chiffre**. Die Information, die Sender und Empfänger geheim halten, ist der **Schlüssel**.

Benutzen beide Seiten denselben Schlüssel, heißt das Verfahren **symmetrisch**. Gibt es getrennte Schlüssel zum Ver- und Entschlüsseln, heißt es **asymmetrisch** (Abschnitt 8.9).

8.2.1 Das Prinzip von Kerckhoffs

Ein Anfängerfehler ist es, die Sicherheit auf die Geheimhaltung des Verfahrens zu gründen. Auguste Kerckhoffs formulierte 1883 die bis heute gültige Gegenregel.

Merke: Prinzip von Kerckhoffs

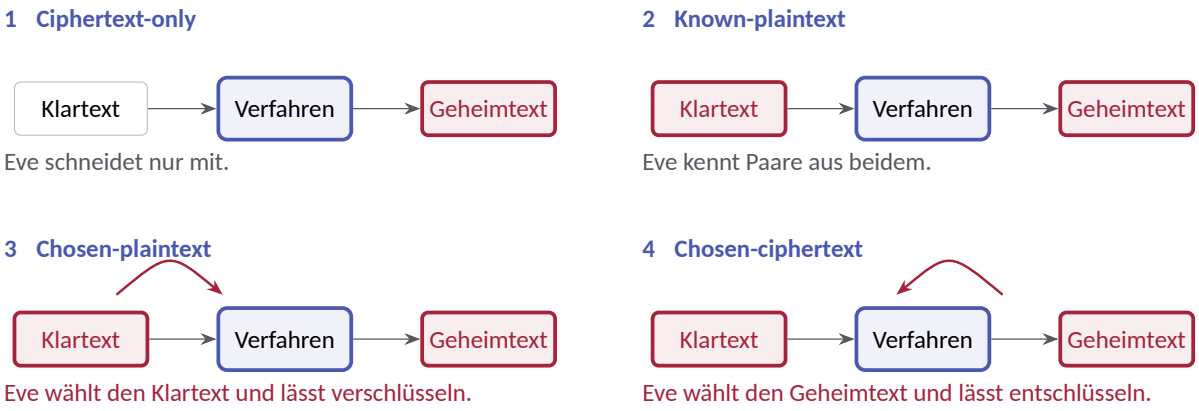
Die Sicherheit eines Verschlüsselungsverfahrens darf allein auf der Geheimhaltung des Schlüssels beruhen, nicht auf der Geheimhaltung des Algorithmus.

Das klingt paradox, ist aber praktisch begründet: Algorithmen werden implementiert, verteilt, disassembliert, von Mitarbeitern weitergetragen. Ein Verfahren, das nur geheim funktioniert, ist genau einmal sicher – bis zum ersten Leck. Ein Schlüssel dagegen lässt sich austauschen, ohne dass alles Übrige neu gebaut werden muss. Moderne Verfahren wie AES sind vollständig veröffentlicht und werden weltweit öffentlich angegriffen; dass sie das aushalten, ist ihr Sicherheitsnachweis.

8.2.2 Angriffsmodelle

Nach Kerckhoffs kennt der Angreifer das Verfahren. Unterschieden wird danach, welches Material ihm darüber hinaus vorliegt.

Gymnasium Ottweiler



Rot = liegt Eve vor. Das *Verfahren* kennt sie nach Kerckhoffs immer, den *Schlüssel* nie.

Die vier Angriffsmodelle. Von oben links nach unten rechts wird Eves Ausgangslage immer besser.

Modell	Der Angreifer hat …	Realistisch, wenn …
Ciphertext-only	nur Geheimtexte	er Datenverkehr mitschneidet
Known-plaintext	Geheimtexte und zugehörige Klartexte	Formulierungen vorhersehbar sind (Briefköpfe, Protokolle)
Chosen-plaintext	er darf Klartexte seiner Wahl verschlüsseln lassen	er Zugang zum Verschlüsselungsgerät hat
Chosen-ciphertext	er darf Geheimtexte seiner Wahl entschlüsseln lassen	ein Server jede Anfrage beantwortet

ZUSATZ

Beide Lehrpläne nennen als Angriffsarten nur Brute-Force, Known-Plaintext und Häufigkeitsanalyse. Die beiden unteren Zeilen der Tabelle — *chosen plaintext* und *chosen ciphertext* — gehen darüber hinaus. Sie stehen hier, weil ohne sie unklar bleibt, woran moderne Verfahren eigentlich gemessen werden.

Ein Verfahren gilt als gut, wenn es selbst im stärksten Modell standhält. Die Verfahren dieses Kapitels scheitern der Reihe nach immer früher, je weiter wir zurückgehen.

Arbeitsauftrag 2

4. Erklären Sie, warum das Prinzip von Kerckhoffs für Open-Source-Software besonders gut passt.
5. Der Klassiker unter den Known-plaintext-Angriffen: Im Zweiten Weltkrieg enthielten viele deutsche Funksprüche das Wort „Wetter“. Erläutern Sie, welchen Vorteil das einem Angreifer verschafft.

8.3 Klassische Verfahren

Klassisch heißen Verfahren, die ohne Computer auskamen. Sie arbeiten mit zwei Grundoperationen, die – in ganz anderem Gewand – bis heute in jeder modernen Blockchiffre stecken.

- **Transposition:** Die Buchstaben bleiben, ihre Reihenfolge ändert sich.
- **Substitution:** Die Reihenfolge bleibt, die Buchstaben werden ersetzt.

8.3.1 Transposition: die Skytala

Die älteste überlieferte Chiffre ist rund 2400 Jahre alt. Die Spartaner wickelten einen schmalen Lederstreifen um einen Stab bestimmter Dicke, die **Skytala**, und schrieben die Nachricht längs des Stabes. Abgewickelt ergab der Streifen eine sinnlose Buchstabenfolge; nur ein Stab desselben Durchmessers machte sie wieder lesbar. Der Schlüssel ist also die Dicke des Stabes.



Eine Skytala. Auf dem aufgewickelten Streifen steht die Nachricht längs des Stabes und ist lesbar. Wickelt man den Streifen ab, stehen die Buchstaben in vertauschter Reihenfolge – der Text ist erst wieder zu entziffern, wenn der Empfänger einen Stab desselben Durchmessers benutzt.

Informatisch ist die Skytala nichts anderes als eine Tabelle: Man schreibt den Text zeilenweise in eine Tabelle mit s Spalten und liest ihn spaltenweise wieder aus.

```
Klartext: ANGRIFFBEIMORGENGRAUEN,  s = 4

A N G R        Auslesen spaltenweise:
I F F B        AIERGE NFIGRN GFMEA RBONU
E I M O
R G E N
G R A U
E N
```

Der Angriff ist einfach: Es gibt nur wenige sinnvolle Spaltenzahlen, man probiert sie durch. Die Skytala scheitert schon im Ciphertext-only-Modell.

8.3.2 Substitution: die Cäsar-Chiffre

Bei der **Cäsar-Chiffre** wird das Alphabet zyklisch um k Stellen verschoben. Nach Z folgt wieder A. Der Schlüssel ist die Zahl k.

```
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
Geheim (k=4): 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 D
```

Rechnet man die Buchstaben in Zahlen um (A = 0, B = 1, ..., Z = 25), wird daraus eine Addition mit Rest:

$$c = (m + k) \bmod 26 \qquad \text{und} \qquad m = (c - k) \bmod 26$$

Dabei ist m der Klartext- und c der Geheimtextbuchstabe. Das „mod 26“ erledigt genau den Übergang von Z zurück zu A.

Weil es nur 25 brauchbare Schlüssel gibt, ist die Cäsar-Chiffre mit Bleistift und Papier in Minuten zu brechen: Man probiert alle durch (**Brute-Force-Angriff**). Ein Computer erledigt das in Mikrosekunden.

8.3.3 Allgemeine monoalphabetische Substitution

Man kann jedem Buchstaben statt der Verschiebung eine beliebige Zuordnung geben – dann ist der Schlüssel eine vollständige Vertauschung des Alphabets. Davon gibt es

$$26! = 403\,291\,461\,126\,605\,635\,584\,000\,000 \approx 4 \cdot 10^{26}$$

Stück. Eine Maschine, die eine Milliarde Schlüssel pro Sekunde prüft, wäre damit rund 13 Milliarden Jahre beschäftigt — ungefähr so lange, wie das Universum alt ist. Trotzdem ist das Verfahren unsicher, und diese Diskrepanz ist die wichtigste Lehre dieses Abschnitts.

Merke

Ein großer Schlüsselraum ist notwendig, aber nicht hinreichend für Sicherheit. Er schützt nur vor Brute Force, nicht vor einem klügeren Angriff.

8.3.4 Kryptoanalyse durch Häufigkeitsanalyse

Der klügere Angriff nutzt die Statistik der Sprache. In deutschen Texten kommen die Buchstaben höchst ungleich vor:

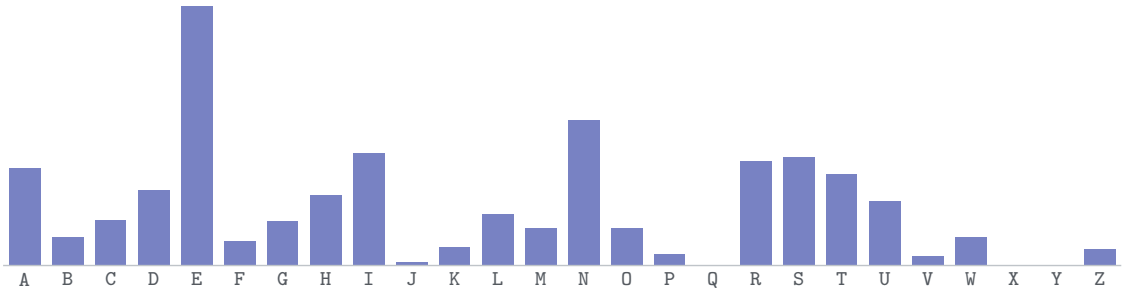
Buchstabe	E	N	I	S	R	A	T	D	H	U	L	C
Anteil %	17,4	9,8	7,6	7,3	7,0	6,5	6,2	5,1	4,8	4,4	3,4	3,1

Allein E und N machen gut ein Viertel jedes Textes aus, die zwölf häufigsten Buchstaben rund 80 %. Eine monoalphabetische Substitution verschiebt diese Verteilung nur, sie zerstört sie nicht: Der häufigste Geheimtextbuchstabe ist mit großer Wahrscheinlichkeit das E.

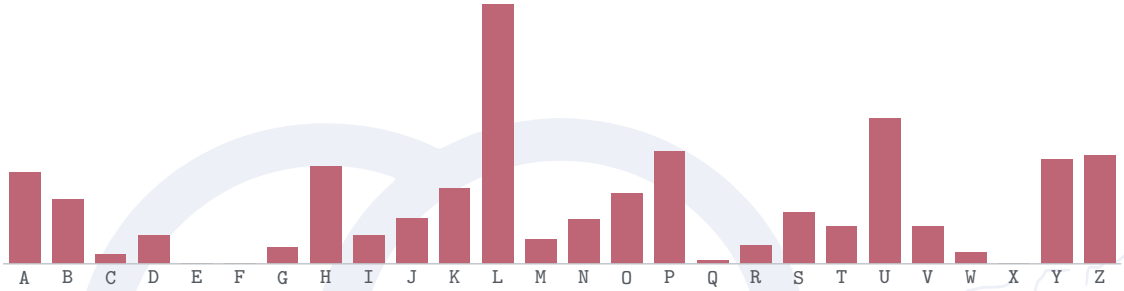
> In der Webfassung steht hier ein Bedienfeld zum Ausprobieren: **Verschlüsselungswerkzeug**.

Probieren Sie es im Bedienfeld aus: Bei Cäsar wandert die Verteilung geschlossen um k Stellen nach rechts — die Form bleibt erkennbar. Schalten Sie dann auf Vigenère um, und die Verteilung flacht ab. Genau deshalb braucht der Angriff auf Vigenère einen zweiten Gedanken, den Kasiski-Test aus dem nächsten Abschnitt.

Deutscher Klartext



Geheimtext nach Cäsar mit $k = 7$



Dieselben Balken, nur um sieben Stellen gewandert. Die Substitution verschiebt die Verteilung — sie zerstört sie nicht. Der höchste Balken ist immer noch das E, es heißt jetzt nur L.

Buchstabenhäufigkeit im Deutschen, darunter dieselbe Verteilung nach einer Cäsar-Chiffre mit $k = 7$.

Das untere Diagramm ist der ganze Angriff: Man sucht den höchsten Balken und weiß, dass er das E sein muss. Daraus ergibt sich der Schlüssel, und der Rest fällt von allein.

Hilfreich sind außerdem **Bigramme** (Buchstabenpaare). Im Deutschen sind ER, EN, CH, DE und EI besonders häufig; CH und SCH verraten sich schnell. Bei kurzen oder ungewöhnlichen Texten versagt die Methode allerdings – ein Text „über den Einfluss von Ozon auf die Zebras im Zentrum von Zaire“ hat eine ganz andere Statistik als die Zeitung von gestern.

Arbeitsauftrag 3

6. Verschlüsseln Sie `INFORMATIK` mit der Cäsar-Chiffre und $k = 7$.
7. Entschlüsseln Sie den Geheimtext `OBXE LITLL UXBF XGMLVAENXLLXEG`. Notieren Sie Ihren Weg.
8. Schreiben Sie ein Programm, das zu einem Text und einem Schlüssel k den Geheimtext liefert, und ein zweites, das zurückrechnet. Buchstaben außerhalb von A–Z sollen unverändert bleiben.
9. Erweitern Sie Ihr Programm so, dass es alle 25 Möglichkeiten ausgibt – der vollständige Brute-Force-Angriff.
10. Schreiben Sie ein Programm, das die Häufigkeit aller Zeichen eines Textes zählt und als Balkendiagramm ausgibt. Testen Sie es an einem längeren deutschen Text.
11. Begründen Sie, warum ein Brute-Force-Angriff auf die allgemeine monoalphabetische Substitution aussichtslos, das Verfahren aber trotzdem unsicher ist.

8.4 Polyalphabetische Verschlüsselung: Vigenère

Die Antwort auf die Häufigkeitsanalyse liegt nahe: Man benutzt nicht ein Geheimalphabet, sondern mehrere im Wechsel. Das bekannteste dieser Verfahren veröffentlichte der französische Diplomat Blaise de Vigenère 1586.

Unter den Klartext wird ein **Schlüsselwort** geschrieben und so oft wiederholt, bis der Text zu Ende ist. Jeder Buchstabe des Schlüsselworts gibt an, um wie viele Stellen der darüberstehende Klartextbuchstabe verschoben wird (A = 0, B = 1, …).

```
Klartext:   K  R  Y  P  T  O  G  R  A  P  H  I  E
Schlüssel:  H  A  L  L  O  H  A  L  L  O  H  A  L
Geheimtext: R  R  J  A  H  V  G  C  L  D  O  I  P
```

Von Hand liest man die Geheimtextbuchstaben im **Vigenère-Quadrat** ab: Die Zeile gibt der Schlüsselbuchstabe vor, die Spalte der Klartextbuchstabe; im Kreuzungspunkt steht der Geheimtextbuchstabe. Für das erste Zeichen des Beispiels sucht man also die Zeile H und darin die Spalte K und findet R.

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

Jede Zeile des Quadrats ist nichts anderes als ein um eine Stelle weiter verschobenes Alphabet – also eine Cäsar-Chiffre. Das Vigenère-Verfahren besteht demnach darin, im Takt des Schlüsselworts zwischen 26 Cäsar-Chiffren hin- und herzuspringen. Genau diese Beobachtung wird uns beim Angriff in Abschnitt 8.4.1 wieder begegnen.

Rechnerisch ist es eine Addition mit Rest, nur dass sich der Summand von Buchstabe zu Buchstabe ändert:

$$c_i = (m_i + s_{i \bmod l}) \bmod 26$$

Dabei ist l die Länge des Schlüsselworts.

Der Gewinn ist deutlich: Gleiche Klartextbuchstaben werden je nach Position verschieden verschlüsselt, die Häufigkeitsverteilung des Geheimtextes wird flach. Drei Jahrhunderte lang galt das Verfahren als unknackbar – „le chiffre indéchiffrable“.

8.4.1 Der Kasiski-Test

1863 zeigte der preußische Major Friedrich Wilhelm Kasiski, wo die Schwäche liegt: Der Schlüssel **wiederholt sich**. Trifft dieselbe Klartextstelle zufällig auf dieselbe Stelle des Schlüsselworts, entsteht dieselbe Geheimtextstelle. Der Abstand solcher Wiederholungen ist deshalb ein Vielfaches der Schlüssellänge.

Daraus ergibt sich der Angriff in drei Schritten:

1. Im Geheimtext wiederholte Buchstabenfolgen (Länge 4 oder 5) suchen und ihre Abstände berechnen.
2. Den größten gemeinsamen Teiler der Abstände bilden – das ist die vermutete Schlüssellänge l. Ein einzelner Abstand genügt nicht, denn er ist nur ein Vielfaches von l.
3. Den Text in l Spalten zerlegen. Jede Spalte ist mit *einem* festen Alphabet verschlüsselt, also eine gewöhnliche Cäsar-Chiffre – und gegen die hilft die Häufigkeitsanalyse aus 8.3.4.

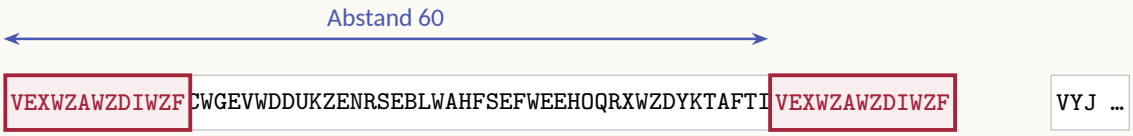
Beispiel: Ein Geheimtext, vier Abstände

Der folgende Text wurde mit einem Schlüsselwort verschlüsselt, dessen Länge wir nicht kennen:

```
  0  VEXWZAWZDIWZFDCWGEVWDDUKZENRSEBLWAHFSEFWEEHOQRXWZDYKTAFTII
 58  LVVEXWZAWZDIWZFVYJECBDGEMKQLNOQRXAQNUUTRCUTTFAQSNKUEBLZULT
116  GCBKFAVWZSUDMTQWDDYFECBDGEMKQLBSFSCWTTXAQNUUTRCUTTMGQIHXMC
174  BAETXWDGLMZDAWPAHCQUHVEOMUTWYJUSNKQIHWGMMWFZOFSIHVQRJJMXCK
```

Vier Folgen kommen doppelt vor:

Folge	Positionen	Abstand
VEXWZAWZDIWZF	0 und 60	60
ECBDGEMKQL	76 und 136	60
XAQNUUTRCUTT	90 und 154	64
OQRX	43 und 87	44



Dieselbe Folge an zwei Stellen. Der Abstand muss ein *Vielfaches* der Schlüssellänge sein — welches, verrät ein einzelner Abstand nicht.

$$\text{ggT}(60,\ 64,\ 60,\ 44) = \mathbf{4}$$

erst mehrere Abstände legen die Schlüssellänge fest

Zwei Fundstellen derselben Folge und ihr Abstand.

Der größte gemeinsame Teiler von 60, 64, 60 und 44 ist **4**. Das Schlüsselwort hat also vier Buchstaben — es war `MAUS`. Mit nur einem Abstand wäre man nicht weitergekommen: 60 allein lässt 1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30 und 60 zu.

Merke

Die Vigenère-Chiffre ist nur so stark wie die Periode ihres Schlüssels. Ein sich wiederholender Schlüssel zerlegt den Text in mehrere einfache Chiffren.

1922 ergänzte William F. Friedman den **Friedman-Test**: Er schätzt die Schlüssellänge über den *Koinzidenzindex*, also die Wahrscheinlichkeit, dass zwei zufällig gezogene Buchstaben eines Textes gleich sind. Für deutschen Klartext liegt dieser Wert bei etwa 0,076, für zufällige Buchstaben bei $1/26 \approx 0{,}038$. Je länger der Schlüssel, desto näher rückt der Geheimtext an den Zufallswert.

Arbeitsauftrag 4

12. Verschlüsseln Sie `SICHERHEIT` mit dem Schlüsselwort `MAUS`.
13. Entschlüsseln Sie `DLGK TKQT` mit dem Schlüsselwort `TAG`.
14. Erklären Sie, warum ein Schlüsselwort mit lauter gleichen Buchstaben (z. B. `AAAA`) die Vigenère-Chiffre auf ein bekanntes Verfahren zurückführt. Auf welches?
15. Schreiben Sie ein Programm zur Ver- und Entschlüsselung nach Vigenère.

8.4.2 Werkstatt: ein Vigenère-Chiffrat knacken

Der folgende Text ist deutsch, mit einem unbekannten Schlüsselwort verschlüsselt und 509 Buchstaben lang. Umlaute stehen als AE/OE/UE, Satzzeichen und Leerzeichen sind entfallen. Die Zahl am Zeilenanfang gibt die Position des ersten Buchstabens der Zeile an – damit lassen sich Abstände ablesen, ohne nachzuzählen.

```
  1  NIWKZ SNPRW DGXDS GWVWB MFEJA URTEE KAWVB TEVMF LCMHK BSLMX
 51  MABWS VOMSV VOZON STHRI LELCF NDORO QKMPD SGLAO PKEMN AQSOI
101  LMFNM CKDIJ DMHTW AARZO SUPOK MMHWT AOODG WVSEL KRAVV KACSA
151  KZNIC SUPDE MCSWT OYZDW AMVOZ RODBO OZNIW TSOVQ EVMKC KRLMM
201  KCMVS CMFXB NEJHW BTOGL LWXBO XLQFC XKLLM FEVNK FIUUB TEVMW
251  SVJED VWCXK LLMES BNEJO WGWRN LMFRI OUXQY UMSTK IFKTI SWLWB
301  XBEMA KSAMH WWXPQ JIWZX BQODJ QURSK SAACS DORGM XPMXT DQURB
351  ODAMK OVKNY ZAPNL EJMAD ASMBI ZBIMH LHWRV RUFLW BBNRW QMXLC
401  EUPRS OORKC URBOI EKZSN PRSBF KKRWA MVOZR ODBWX JECZA LKJON
451  XWDQM XUFLT OADIE ULOIE SVMJO VKBKB SOVNE FLAOT KEFOW NMCSU
501  PDEMC SWTK
```

Dass die einfache Häufigkeitsanalyse hier nicht weiterhilft, zeigt schon ein Blick auf die Verteilung: Der häufigste Buchstabe des Chiffrats ist M mit 7,7 %, gefolgt von O mit 7,3 % und gleich drei Buchstaben mit 6,1 %. Im Deutschen läge das E dagegen bei 17,4 %. Die Verteilung ist flach – aber, wie Kasiski gezeigt hat, nicht strukturlos.

16 Suchen Sie im Chiffrat Buchstabenfolgen der Länge 5, die zweimal vorkommen, und notieren Sie mindestens vier Funde mit ihren Positionen und dem Abstand. Bestimmen Sie zu jedem Abstand *alle* Teiler und markieren Sie die Zahl, die in jeder Zeile vorkommt. Begründen Sie, warum ein einzelner Fund nicht genügt: Welche Schlüssellängen ließe Ihr kleinster Abstand noch offen?

Legen Sie dafür im Heft eine Tabelle mit den Spalten *Fundstelle*, *Position 1*, *Position 2*, *Abstand* und *alle Teiler des Abstands* an.

17 Zerlegen Sie das Chiffrat in m Spalten und bestimmen Sie für jede Spalte den häufigsten Buchstaben. Nehmen Sie an, dass dies das verschlüsselte E ist, und berechnen Sie daraus den Schlüsselbuchstaben. Arbeiten Sie arbeitsteilig – je Gruppe eine Spalte.

Tabellenspalten: *Spalte*, *Anzahl Buchstaben*, *häufigster Buchstabe*, *Verschiebung*, *Schlüsselbuchstabe*.

18 Notieren Sie das Schlüsselwort. Fällt Ihnen an seinen Buchstaben etwas auf, das Ihre Rechnung bestätigt? Entschlüsseln Sie anschließend die ersten beiden Zeilen von Hand und setzen Sie Wortgrenzen. Der Klartext verrät, wem Sie dieses Verfahren verdanken.

19 Automatisieren Sie den Angriff im Notebook *vigenere-info12.ipynb*: Häufigkeiten zählen, Wiederholungen und Abstände finden, ggT bilden, Spalten auswerten, entschlüsseln.

20 *Für Schnelle:* Lassen Sie Ihr Programm alle Schlüssellängen von 1 bis 12 durchprobieren und jedes Ergebnis über die Abweichung von der deutschen Buchstabenverteilung bewerten (χ^2). Der Angriff läuft dann ganz ohne Vorwissen: Bei diesem Text schneidet die richtige Länge rund zwanzigmal besser ab als die nächstbeste. Beachten Sie dabei eine Falle — auch jedes *Vielfache* der richtigen Länge liefert einen guten Wert. Warum?

8.5 Homophone Verschlüsselung

ZUSATZ

> ZUSATZ
>
> Die Abschnitte 8.5 und 8.6 stehen in keinem der beiden Lehrpläne. Sie schließen aber die Lücke zwischen Vigenère und den modernen Verfahren: erst die Statistik verwischen, dann die Wiederholung des Schlüssels abschaffen.

Ein anderer Weg, die Statistik zu verwischen: Jedem Klartextbuchstaben wird nicht ein Zeichen zugeordnet, sondern eine *Menge* von Zeichen – und zwar so viele, wie es seiner Häufigkeit entspricht. Beim Verschlüsseln wählt man daraus zufällig aus.

```
A: 10, 21, 52, 59, 71, 84
B: 20, 34
C: 06, 28, 80
E: 09, 18, 33, 38, 40, 42, 53, 54, 55, 60, 66, 75, 85, 86, 92, 93, 99
```

Weil dem E siebzehn Zeichen zur Verfügung stehen, dem B nur zwei, kommt am Ende jedes Zeichen etwa gleich oft vor: Die Häufigkeitsanalyse läuft ins Leere. Der Preis ist ein längerer Geheimtext und eine große Tabelle als Schlüssel. Sicher ist auch dieses Verfahren nicht – Bigramme und Wortstrukturen bleiben erhalten und wurden historisch erfolgreich ausgenutzt.

Gymnasium Ottweiler

8.6 Das One-Time-Pad

ZUSATZ

Aus den bisherigen Angriffen lassen sich zwei Forderungen ableiten: Der Schlüssel darf sich nicht wiederholen, er muss also so lang sein wie die Nachricht, und er darf nur ein einziges Mal benutzt werden. Genau das leistet das **One-Time-Pad** (Vernam-Chiffre, 1917).

Der Schlüssel ist eine echt zufällige Buchstaben- oder Bitfolge, von der Sender und Empfänger je ein identisches Exemplar besitzen. Jedes Zeichen wird mit genau einem Schlüsselzeichen verknüpft, danach wird der verbrauchte Teil vernichtet.

Das One-Time-Pad ist das einzige Verfahren, dessen Sicherheit **bewiesen** ist (Shannon, 1949). Der Grund lässt sich mit den Werkzeugen dieses Kapitels formulieren: Zu einem gegebenen Geheimtext existiert für *jeden* denkbaren Klartext gleicher Länge ein passender Schlüssel. Der Kasiski-Test findet keine Periode, die Häufigkeitsanalyse keine Spalten. Der Geheimtext verrät über den Klartext nichts außer dessen Länge.

Warum benutzen wir es dann nicht überall? Wegen der Praxis:

- Der Schlüssel ist so lang wie alle Nachrichten zusammen und muss vorab sicher ausgetauscht werden. Wer das kann, könnte auch die Nachricht selbst sicher austauschen.
- Er muss echt zufällig sein; Pseudozufallszahlen aus einem Programm genügen nicht.
- Jede Wiederverwendung bricht das Verfahren sofort. Die sowjetischen Funksprüche des Venona-Projekts wurden genau deshalb entziffert: Schlüsselblöcke waren aus Zeitnot doppelt verwendet worden.

Arbeitsauftrag 5

21. Begründen Sie, warum aus einem One-Time-Pad-Geheimtext prinzipiell kein Klartext ableitbar ist, obwohl der Angreifer das Verfahren kennt.
22. Zwei Nachrichten wurden versehentlich mit demselben One-Time-Pad verschlüsselt. Zeigen Sie: Aus c_1 und c_2 lässt sich etwas berechnen, das den Schlüssel nicht mehr enthält. Was?
23. Erklären Sie, warum das One-Time-Pad für den Betrieb einer Webseite ungeeignet ist.

Gymnasium Ottweiler

8.7 Moderne symmetrische Verfahren

Heutige symmetrische Verfahren verschlüsseln keine Buchstaben, sondern Bitblöcke fester Länge (**Blockchiffren**). Der Standard ist seit 2001 **AES** (Advanced Encryption Standard) mit Schlüssellängen von 128, 192 oder 256 Bit. AES wendet auf jeden 128-Bit-Block mehrere Runden aus Substitution und Transposition an – dieselben zwei Ideen wie in Abschnitt 8.3, nur auf Bits und vielfach verschachtelt.

Bei 128 Bit gibt es $2^{128} \approx 3{,}4 \cdot 10^{38}$ Schlüssel. Eine Maschine, die eine Billion Schlüssel pro Sekunde prüft, wäre damit rund 10^{19} Jahre beschäftigt. Brute Force ist hier endgültig kein Weg mehr.

Ein Problem bleibt aber, und es ist kein mathematisches: Sender und Empfänger brauchen denselben geheimen Schlüssel. Bei n Teilnehmern, die alle paarweise vertraulich kommunizieren wollen, sind das

$$\frac{n \cdot (n-1)}{2}$$

Schlüssel – bei 1000 Teilnehmern also fast eine halbe Million, und jeder einzelne muss sicher ausgetauscht werden, bevor die erste Nachricht fließt. Das ist das **Schlüsselverteilungsproblem**.

Arbeitsauftrag 6

24. Berechnen Sie die Zahl der nötigen Schlüssel für eine Schule mit 900 Personen.
25. Ein Angreifer kann eine Milliarde AES-Schlüssel pro Sekunde testen. Schätzen Sie die Dauer eines Brute-Force-Angriffs auf einen 128-Bit-Schlüssel und vergleichen Sie mit dem Alter des Universums (rund $1{,}4 \cdot 10^{10}$ Jahre). Vergleichen Sie auch mit Abschnitt 8.3.3: Warum ist der Unterschied zur monoalphabetischen Substitution so gewaltig, obwohl beide Schlüsselräume „riesig" heißen?

Gymnasium Ottweiler

8.8 Schlüsseltausch nach Diffie und Hellman

Das Schlüsselverteilungsproblem sieht aus wie ein Henne-Ei-Problem: Um sicher zu reden, braucht man einen gemeinsamen Schlüssel — und um den auszutauschen, müsste man schon sicher reden können. 1976 zeigten **Whitfield Diffie** und **Martin Hellman**, dass dieser Schluss falsch ist. Zwei Partner können sich auf einen gemeinsamen geheimen Schlüssel einigen, indem sie **ausschließlich öffentlich** miteinander sprechen. Niemand muss das Geheimnis jemals übertragen, weil es erst beim Rechnen entsteht.

Leitidee: Die Idee in einem Bild

Alice und Bob mischen Farben. Beide starten mit demselben, allen bekannten Gelb. Alice gibt heimlich etwas Rot dazu, Bob heimlich etwas Blau; die Mischungen tauschen sie offen aus. Nun gibt Alice ihr Rot in Bobs Mischung, Bob sein Blau in Alices — beide halten am Ende denselben Braunton in der Hand. Ein Beobachter hat Gelb, Orange und Grün gesehen und kommt damit nicht weiter: Mischen ist leicht, Entmischen ist schwer.

Die Rolle des Mischens übernimmt in der Mathematik das **Potenzieren mit Rest**. Öffentlich vereinbart werden eine Primzahl p und eine Zahl g mit $1 < g < p$, die **Basis**. Beide Werte dürfen alle kennen.

Schritt	Alice	Bob
1	wählt geheim a	wählt geheim b
2	berechnet $A = g^a \bmod p$	berechnet $B = g^b \bmod p$
3	sendet A an Bob	sendet B an Alice
4	berechnet $K = B^a \bmod p$	berechnet $K = A^b \bmod p$

Dass am Ende wirklich dieselbe Zahl herauskommt, ist nichts als ein Potenzgesetz:

$$K = B^a = \left(g^b\right)^a = g^{b \cdot a} = g^{a \cdot b} = (g^a)^b = A^b \pmod p$$

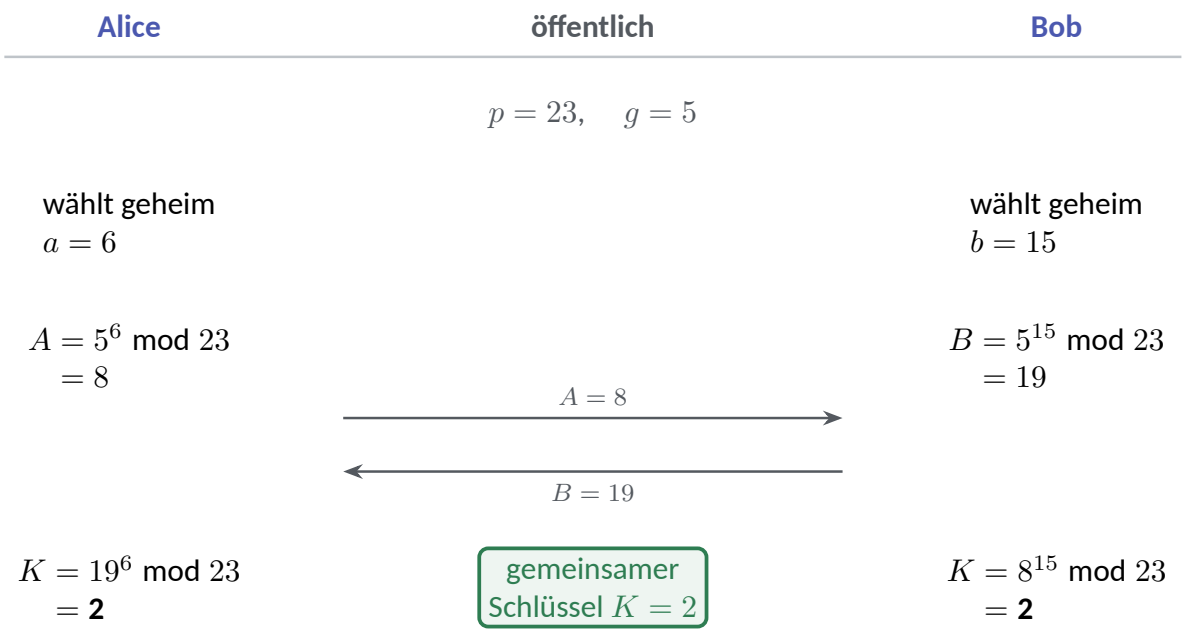
Beispiel: Durchgerechnet mit kleinen Zahlen

Öffentlich: $p = 23$ und $g = 5$.

- Alice wählt $a = 6$ und berechnet $A = 5^6 \bmod 23 = 15\,625 \bmod 23 = 8$.
- Bob wählt $b = 15$ und berechnet $B = 5^{15} \bmod 23 = 19$.
- Sie tauschen $A = 8$ und $B = 19$ offen aus.
- Alice rechnet $K = 19^6 \bmod 23 = 2$.
- Bob rechnet $K = 8^{15} \bmod 23 = 2$.

Beide haben $K = 2$, ohne dass die 2 jemals über die Leitung ging. Die Probe: $5^{6 \cdot 15} \bmod 23 = 5^{90} \bmod 23 = 2$.

Rechnen Sie $5^{15} \bmod 23$ nicht mit dem Taschenrechner aus — 5^{15} hat elf Stellen. Nehmen Sie nach jedem Schritt den Rest: $5^2 = 25 \equiv 2$, also $5^4 \equiv 2^2 = 4$, $5^8 \equiv 4^2 = 16$ und damit $5^{15} = 5^8 \cdot 5^4 \cdot 5^2 \cdot 5^1 \equiv 16 \cdot 4 \cdot 2 \cdot 5 = 640 \equiv 19 \pmod{23}$. In Python genügt `pow(5, 15, 23)`.



Übertragen werden nur p, g, A und B. Wer alles vier mitliest, müsste daraus a oder b zurückrechnen — den *diskreten Logarithmus*. Für kleine Zahlen geht das durch Probieren, für Zahlen mit Hunderten Stellen nicht.

Der Schlüsseltausch nach Diffie und Hellman.

Warum ist das sicher? Ein Lauscher kennt p, g, A und B. Um K zu berechnen, bräuchte er einen der beiden geheimen Exponenten, müsste also aus $A = g^a \bmod p$ die Zahl a zurückgewinnen. Diese Aufgabe heißt **diskreter Logarithmus**. Bei $p = 23$ löst man sie in zwanzig Versuchen; bei einer Primzahl mit 600 Stellen ist kein Verfahren bekannt, das in vertretbarer Zeit zum Ziel kommt. Potenzieren mit Rest ist also eine Einwegfunktion — leicht vorwärts, praktisch unmöglich rückwärts.

Merke

Diffie-Hellman ist **kein Verschlüsselungsverfahren**. Es überträgt keine Nachricht, sondern erzeugt bei beiden Partnern dieselbe Zufallszahl. Verschlüsselt wird danach symmetrisch, etwa mit AES — der ausgehandelte Wert ist der Sitzungsschlüssel. Genau so arbeitet heute jede HTTPS-Verbindung.

Stolperstein

Der Schlüssel entsteht bei jedem Verbindungsaufbau neu und wird danach verworfen. Das hat einen Namen und einen Zweck: **Perfect Forward Secrecy**. Wer den heutigen Datenverkehr mitschneidet und in zehn Jahren den privaten Schlüssel des Servers erbeutet, kann damit nichts anfangen — die Sitzungsschlüssel von damals stehen nirgends mehr.

LK

Der Mittelsmann. Diffie-Hellman sichert zu, dass am Ende *zwei* Partner denselben Schlüssel besitzen. Es sichert nicht zu, **wer** diese beiden sind. Mallory kann sich in die Leitung setzen und mit jeder Seite einen eigenen Schlüsseltausch durchführen:

1. Mallory wählt selbst ein geheimes $m = 3$ und berechnet $M = 5^3 \bmod 23 = 10$.
2. Alice sendet $A = 8$ los. Mallory fängt es ab und schickt Bob stattdessen $M = 10$.
3. Bob sendet $B = 19$. Mallory fängt auch das ab und schickt Alice $M = 10$.
4. Alice rechnet $K_1 = 10^6 \bmod 23 = 6$ — und Mallory rechnet $8^3 \bmod 23 = 6$.
5. Bob rechnet $K_2 = 10^{15} \bmod 23 = 5$ — und Mallory rechnet $19^3 \bmod 23 = 5$.

Mallory hat nun **zwei** Schlüssel: 6 für die Strecke zu Alice, 5 für die Strecke zu Bob. Er entschlüsselt jede Nachricht, liest sie, verschlüsselt sie neu und reicht sie weiter. Alice und Bob bemerken nichts, denn beide Verbindungen sind kryptographisch einwandfrei. Wie sich das abwehren lässt, steht im nächsten Abschnitt.

Arbeitsauftrag 7

26. Führen Sie den Schlüsseltausch mit $p = 13$, $g = 2$, $a = 5$ und $b = 9$ durch. Geben Sie A, B und K an und prüfen Sie, dass beide Seiten dasselbe erhalten.
27. Begründen Sie mit dem Potenzgesetz, warum B^a und A^b denselben Rest lassen müssen.
28. Ein Lauscher kennt $p = 23$, $g = 5$ und $A = 8$. Bestimmen Sie a durch Probieren und notieren Sie, wie viele Versuche Sie gebraucht haben. Was sagt das über die Sicherheit kleiner Primzahlen?
29. Warum darf a nicht 1 sein? Was passiert, wenn Alice und Bob zufällig denselben geheimen Wert wählen?
30. Berechnen Sie $7^{20} \bmod 11$ von Hand, indem Sie nach jedem Schritt den Rest nehmen. Vergleichen Sie mit `pow(7, 20, 11)`.
31. Erklären Sie in drei Sätzen, warum Diffie-Hellman das Schlüsselverteilungsproblem löst, das Problem der Echtheit eines Gegenübers aber nicht.

Gymnasium Ottweiler

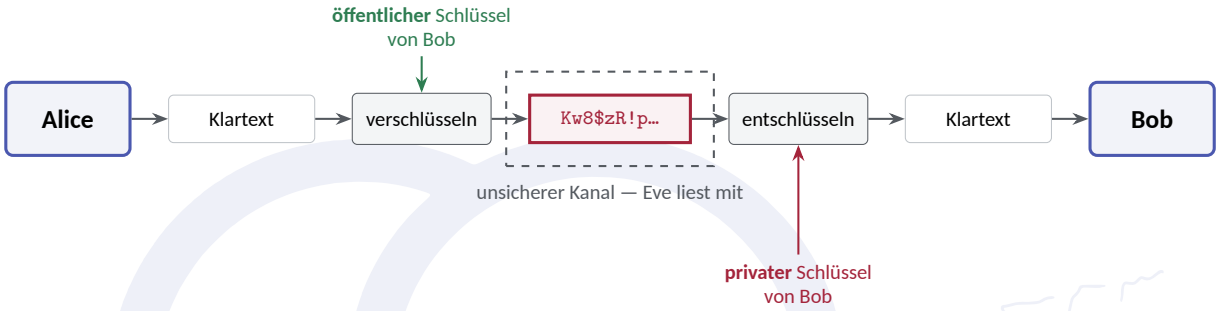
8.9 Asymmetrische Verfahren

1976 schlugen Whitfield Diffie und Martin Hellman eine Lösung vor, die zunächst unmöglich klingt: Jeder Teilnehmer besitzt **zwei** Schlüssel.

- Der **öffentliche Schlüssel** (public key) wird veröffentlicht. Mit ihm kann jeder eine Nachricht an den Besitzer verschlüsseln.
- Der **private Schlüssel** (private key) bleibt geheim und verlässt den Besitzer nie. Nur mit ihm lässt sich wieder entschlüsseln.

Damit verschwindet das Schlüsselverteilungsproblem: Es muss nie ein Geheimnis übertragen werden. Bei n Teilnehmern genügen n Schlüsselpaare.

Die Metapher dazu ist ein Briefkasten: Den Einwurfschlitz kann jeder benutzen, herausholen kann nur, wer den Schlüssel hat.



Bob veröffentlicht den grünen Schlüssel. Jeder darf damit *ver*schlüsseln — *ent*schlüsseln kann nur er, mit dem roten. Ein Geheimnis wird nie übertragen.

Asymmetrische Verschlüsselung. Verschlüsselt wird mit dem Schlüssel des *Empfängers*, nicht des Absenders.

Mathematisch braucht man dafür **Einwegfunktionen**: Berechnungen, die in eine Richtung leicht, in die andere praktisch unmöglich sind. Das Standardbeispiel ist die Multiplikation von Primzahlen.

$$\underbrace{7919 \cdot 6841 = 54\,173\,879}_{\text{leicht}} \qquad \underbrace{54\,173\,879 = ? \cdot ?}_{\text{schwer}}$$

Für die Verschlüsselung braucht man etwas mehr, nämlich eine **Falltürfunktion**: eine Einwegfunktion, die sich doch umkehren lässt, wenn man eine Zusatzinformation besitzt – die Falltür ist der private Schlüssel.

Merke

Asymmetrische Verfahren beruhen nicht darauf, dass ein Angreifer den privaten Schlüssel nicht berechnen *kann*, sondern darauf, dass die Berechnung zu lange dauert. Sicherheit ist hier eine Frage des Aufwands, nicht der Unmöglichkeit.

8.9.1 Der Haken: Man-in-the-Middle

LK

LK

Den Man-in-the-Middle-Angriff verlangt ausdrücklich nur der LK-Lehrplan („beschreiben den Man-in-the-middle-Angriff auf das Diffie-Hellman-Verfahren"). Dass die Lücke existiert, sollte aber auch im Grundkurs gesagt werden — sonst bleibt der falsche Eindruck zurück, Verschlüsselung allein genüge.

Oben stand, das Schlüsselverteilungsproblem verschwinde. Das stimmt nur halb. Übertragen wird zwar kein Geheimnis mehr — aber Alice muss immer noch wissen, dass der Schlüssel, den sie für Bobs öffentlichen hält, wirklich Bob gehört.

Genau hier setzt der **Man-in-the-Middle-Angriff** an (auch *Mittelsmann*-Angriff). Der Angreifer — traditionell **Mallory** genannt, von *malicious* — stellt sich nicht neben die Leitung, sondern **hinein**.



Beide Verbindungen sind einwandfrei verschlüsselt. Nur verschlüsseln sie zu *Mallory* statt zueinander. Die Mathematik ist in Ordnung — die Zuordnung von Schlüssel zu Person ist es nicht.

Man-in-the-Middle. Mallory führt zwei einwandfrei verschlüsselte Gespräche und übersetzt dazwischen.

Der Ablauf:

1. Alice fragt nach Bobs öffentlichem Schlüssel. Mallory fängt die Antwort ab und schickt stattdessen **seinen eigenen**.
2. Alice verschlüsselt mit diesem Schlüssel und schickt los. Mallory kann entschlüsseln — es ist ja sein Schlüssel.
3. Mallory liest mit, ändert nach Belieben, verschlüsselt mit **Bobs echtem** Schlüssel neu und schickt weiter.
4. Bob antwortet, und dasselbe geschieht in der Gegenrichtung.

Beide merken nichts. Beide Verbindungen sind kryptographisch einwandfrei.

Merke

Ein Man-in-the-Middle bricht die Verschlüsselung nicht — er unterläuft sie. Die Mathematik bleibt unangetastet; was fehlt, ist die Zuordnung von **Schlüssel zu Person**.

Deshalb löst Kryptographie allein das Problem nie ganz. Es braucht einen zweiten Weg, über den sich ein Schlüssel überprüfen lässt: persönliche Übergabe, ein Fingerabdruck über das Telefon — oder eine dritte Stelle, der beide trauen. Das sind die **Zertifikate** aus Abschnitt 8.11.

Stolperstein

Es ist derselbe Angriff, den Sie in Abschnitt 8.8 schon am Schlüsseltausch gesehen haben, und er hat denselben Grund: Beide Verfahren stellen sicher, dass am Ende *zwei* Partner einen Schlüssel teilen — aber nicht, **wer** diese beiden sind. Die Mathematik kann diese Lücke nicht schließen; dafür braucht es die Zertifikate aus Abschnitt 8.11.

Arbeitsauftrag 8

32. Erklären Sie, warum ein Man-in-the-Middle-Angriff kein Widerspruch zu der Aussage ist, RSA sei sicher.
33. Sie bekommen den öffentlichen Schlüssel einer Mitschülerin per E-Mail. Nennen Sie zwei Wege, auf denen Sie prüfen können, ob er wirklich von ihr stammt.
34. Begründen Sie, warum der Angriff auf einem Flughafen-WLAN leichter ist als im Schulnetz.

Gymnasium Ottweiler

8.10 Das RSA-Verfahren

1977 fanden Ronald Rivest, Adi Shamir und Leonard Adleman eine solche Falltürfunktion. Das nach ihnen benannte **RSA-Verfahren** wird bis heute eingesetzt.

8.10.1 Vorbereitung: Rechnen mit Resten

a mod n ist der Rest, den a bei Division durch n lässt. Beispiele: 17 mod $5 = 2$, 24 mod $6 = 0$, -23 mod $75 = 52$ (denn $-23 = (-1) \cdot 75 + 52$).

Die **eulersche Phi-Funktion** $\varphi(n)$ gibt an, wie viele Zahlen t mit $1 \leq t < n$ teilerfremd zu n sind:

n	**teilerfremde** t	$\varphi(n)$
5	1, 2, 3, 4	4
6	1, 5	2
15	1, 2, 4, 7, 8, 11, 13, 14	8

Für RSA genügen zwei Regeln:

- Ist p eine Primzahl, so ist $\varphi(p) = p - 1$.
- Sind $p \neq q$ Primzahlen und $n = p \cdot q$, so ist $\varphi(n) = (p-1)(q-1)$.

Der **Satz von Euler** besagt: Sind m und n teilerfremd, so gilt $m^{\varphi(n)}$ mod $n = 1$. Daraus folgt die Idee von RSA: Man zerlegt den Exponenten $\varphi(n)$ geschickt in zwei Faktoren – einen zum Verschlüsseln, einen zum Entschlüsseln.

Arbeitsauftrag 9

35. Rechnen Sie: 24 mod 5, 232 mod 7, $3\,492\,392$ mod 2.
36. Bestimmen Sie $\varphi(11)$, $\varphi(35)$ und $\varphi(77)$. Prüfen Sie bei $\varphi(35)$ und $\varphi(77)$ die zweite Regel nach, indem Sie zusätzlich abzählen.

8.10.2 Schlüsselerzeugung

1. Zwei verschiedene Primzahlen p und q wählen.
2. $n = p \cdot q$ berechnen.
3. $\varphi(n) = (p-1)(q-1)$ berechnen.
4. Ein e mit $1 < e < \varphi(n)$ wählen, das teilerfremd zu $\varphi(n)$ ist.
5. Das d bestimmen mit $e \cdot d$ mod $\varphi(n) = 1$ (erweiterter euklidischer Algorithmus).

> LK
>
> Schritt 5 ist LK-Stoff: Der GK-Lehrplan verlangt das RSA-Verfahren ausdrücklich *„ohne die Berechnung des privaten Schlüssels"*, der LK dagegen fordert den euklidischen und den erweiterten euklidischen Algorithmus. Im Grundkurs bekommen Sie d also vorgegeben und prüfen nur die Bedingung $e \cdot d$ mod $\varphi(n) = 1$ nach — rechnen Sie Schritt 5 trotzdem einmal mit, er ist der schönste Teil.

Öffentlicher Schlüssel: (e, n). Privater Schlüssel: (d, n). Die Zahlen p, q und $\varphi(n)$ werden vernichtet – wer sie kennt, kann d berechnen.

Verschlüsseln: $c = m^e$ mod n Entschlüsseln: $m = c^d$ mod n

8.10.3 Durchgerechnetes Beispiel

Wir wählen $p = 7$ und $q = 13$.

1. $n = 7 \cdot 13 = 91$
2. $\varphi(n) = 6 \cdot 12 = 72$
3. Wahl von $e = 5$. Probe: ggT$(5, 72) = 1$, also teilerfremd.
4. Gesucht ist d mit $5d$ mod $72 = 1$. Euklidischer Algorithmus: $72 = 14 \cdot 5 + 2$ und $5 = 2 \cdot 2 + 1$, der ggT ist also 1.
5. Rückwärts eingesetzt: $1 = 5 - 2 \cdot 2 = 5 - 2 \cdot (72 - 14 \cdot 5) = 29 \cdot 5 - 2 \cdot 72$. Also ist $d = 29$. Probe: $5 \cdot 29 = 145 = 2 \cdot 72 + 1$.

Öffentlicher Schlüssel $(5, 91)$, privater Schlüssel $(29, 91)$.

Verschlüsselung der Nachricht $m = 3$:

$$c = 3^5 \text{ mod } 91 = 243 \text{ mod } 91 = 61$$

Entschlüsselung:

$$m = 61^{29} \text{ mod } 91 = 3$$

> In der Webfassung steht hier ein Bedienfeld zum Ausprobieren: **RSA-Rechner**.

Im Bedienfeld können Sie p und q verändern. Probieren Sie größere Primzahlen und beobachten Sie zweierlei: Die Rechnung bleibt gleich schnell, der Angriff durch Faktorisieren wird dagegen rasch mühsam.

Die zweite Rechnung führt man nicht mit dem Taschenrechner aus – 61^{29} hat 52 Stellen. Man rechnet **modular**, also nach jedem Schritt den Rest, oder nutzt die schnelle Exponentiation:

```
1 # Modulare Exponentiation - in Python eingebaut
2 print(pow(3, 5, 91))     # 61   Verschluesseln
3 print(pow(61, 29, 91))   # 3    Entschluesseln
```

8.10.4 Sicherheit von RSA

Die Sicherheit steht und fällt mit dem **Faktorisierungsproblem**: Wer n in p und q zerlegen kann, berechnet $\varphi(n)$ und daraus d – RSA ist gebrochen. Ein schnelles Faktorisierungsverfahren ist bis heute nicht bekannt, aber es ist auch nicht bewiesen, dass es keines gibt.

In der Praxis benutzt man deshalb sehr große Zahlen: Schlüssel unter 2048 Bit gelten als überholt, empfohlen werden heute 3072 Bit oder mehr. Unser $n = 91$ ist 7 Bit lang.

Ein Nachtrag, den ältere Darstellungen noch nicht enthalten: Ein hinreichend großer Quantencomputer könnte mit dem Algorithmus von Peter Shor (1994) in Polynomialzeit faktorisieren und damit RSA brechen. Solche Rechner existieren bisher nicht in nutzbarer Größe. Trotzdem werden seit 2024 schrittweise **post-quanten-sichere** Verfahren eingeführt, die auf anderen mathematischen Problemen beruhen; Messenger und Browser setzen sie bereits parallel zu RSA ein.

Arbeitsauftrag 10

37. Führen Sie die Schlüsselerzeugung für $p = 11$ und $q = 13$ mit $e = 7$ vollständig durch und bestimmen Sie d über den erweiterten euklidischen Algorithmus.
38. Verschlüsseln Sie mit Ihrem Schlüssel aus Aufgabe 37 die Zahl $m = 9$ und entschlüsseln Sie das Ergebnis wieder. Kontrollieren Sie mit `pow(...)`.
39. Erklären Sie, warum p und q nach der Schlüsselerzeugung gelöscht werden müssen.
40. Warum darf m nicht größer als n sein? Was folgt daraus für längere Nachrichten?
41. Schreiben Sie ein Programm, das zu gegebenen p, q und e den privaten Schlüssel d berechnet und einen Text zeichenweise ver- und entschlüsselt.

8.11 Signaturen, Hashfunktionen und die Praxis

Verschlüsselung schützt die Vertraulichkeit. Für Integrität und Authentizität kommen zwei weitere Bausteine hinzu.

Hashfunktionen berechnen aus einer beliebig langen Nachricht einen kurzen Wert fester Länge, den Fingerabdruck. Sie sind Einwegfunktionen: Aus dem Hashwert lässt sich die Nachricht nicht zurückgewinnen, und es soll praktisch unmöglich sein, zwei Nachrichten mit gleichem Hashwert zu finden (*Kollision*). Ändert sich ein einziges Bit der Nachricht, ändert sich der Hashwert vollständig. Aktuell ist SHA-256; ältere Verfahren wie MD5 und SHA-1 sind gebrochen – für sie lassen sich Kollisionen gezielt erzeugen – und dürfen für Sicherheitszwecke nicht mehr verwendet werden.

> LK
>
> Alles Weitere in diesem Abschnitt — digitale Signatur, Zertifikate und hybride Verfahren — steht nur im LK-Lehrplan („erläutern die Idee der digitalen Signatur anhand des RSA-Verfahrens", „erläutern die Idee hybrider Verschlüsselungsverfahren"). Die Hashfunktion oben gehört dagegen zum Schutzziel *Integrität* und damit in beide Kurse.

Digitale Signaturen drehen die asymmetrische Idee um. Der Absender verschlüsselt den Hashwert seiner Nachricht mit seinem *privaten* Schlüssel. Jeder kann diesen Wert mit dem *öffentlichen* Schlüssel prüfen und mit dem selbst berechneten Hashwert vergleichen. Stimmen beide überein, stammt die Nachricht vom Besitzer des privaten Schlüssels und wurde nicht verändert – Authentizität und Integrität in einem Schritt.

Offen bleibt: Woher weiß ich, dass ein öffentlicher Schlüssel wirklich zu der Person gehört, die draufsteht? Diese Lücke schließen **Zertifikate**. Eine Zertifizierungsstelle bestätigt mit ihrer eigenen Signatur die Zuordnung von Schlüssel und Inhaber. Ihr Zertifikat ist wiederum signiert – bis hinauf zu Wurzelzertifikaten, die im Betriebssystem und im Browser hinterlegt sind.

Hybride Verfahren verbinden schließlich beide Welten, denn asymmetrische Verfahren sind um Größenordnungen langsamer als symmetrische. Beim Aufruf einer Webseite über HTTPS läuft deshalb Folgendes ab:

1. Der Server sendet sein Zertifikat mit dem öffentlichen Schlüssel.
2. Der Browser prüft die Signaturkette bis zu einem Wurzelzertifikat.
3. Beide handeln asymmetrisch einen gemeinsamen, zufälligen Sitzungsschlüssel aus.
4. Die eigentlichen Daten werden symmetrisch mit diesem Sitzungsschlüssel verschlüsselt (AES).
5. Nach der Sitzung wird der Schlüssel verworfen.

Merke

Asymmetrisch wird der Schlüssel ausgetauscht, symmetrisch werden die Daten verschlüsselt. Das Schloss im Browser bedeutet genau das – und nicht, dass die Seite vertrauenswürdig ist.

Arbeitsauftrag 11

42. Erklären Sie, warum beim Signieren der Hashwert und nicht die ganze Nachricht verschlüsselt wird.
43. Ein Angreifer findet zwei verschiedene Verträge mit identischem MD5-Hashwert. Beschreiben Sie, welchen Schaden er damit anrichten kann.
44. Rufen Sie im Browser das Zertifikat einer beliebigen Webseite auf. Notieren Sie Aussteller, Gültigkeitsdauer und Schlüssellänge und ordnen Sie die Angaben diesem Kapitel zu.
45. Begründen Sie, warum das Schloss-Symbol nichts über die Seriosität des Anbieters aussagt.

8.12 Exkurs: Blockchain, Kryptowährungen und Mining

ZUSATZ

> ZUSATZ
>
> Ein Exkurs im Wortsinn: Kein Lehrplan verlangt ihn. Er benutzt ausschließlich Bausteine, die Sie schon kennen — Hashfunktion und Signatur — und zeigt, was sich daraus bauen lässt.

Über kaum eine Technik ist so viel geschrieben worden wie über die Blockchain, und kaum eine ist so einfach. Sie besteht fast nur aus dem Baustein des vorigen Abschnitts: der Hashfunktion. Wer verstanden hat, was SHA-256 tut, versteht in zwanzig Minuten eine Blockchain.

8.12.1 Das Problem: ein Kassenbuch ohne Kassenwart

Eine Überweisung ist nichts Körperliches. Sie ist eine Zeile in einem Buch: *Lena zahlt 10 an Jonas*. Damit das funktioniert, muss jemand das Buch führen, dem alle vertrauen — eine Bank. Sie verhindert, dass Lena dieselben 10 Euro zweimal ausgibt, und sie verhindert, dass jemand nachträglich Zeilen ändert.

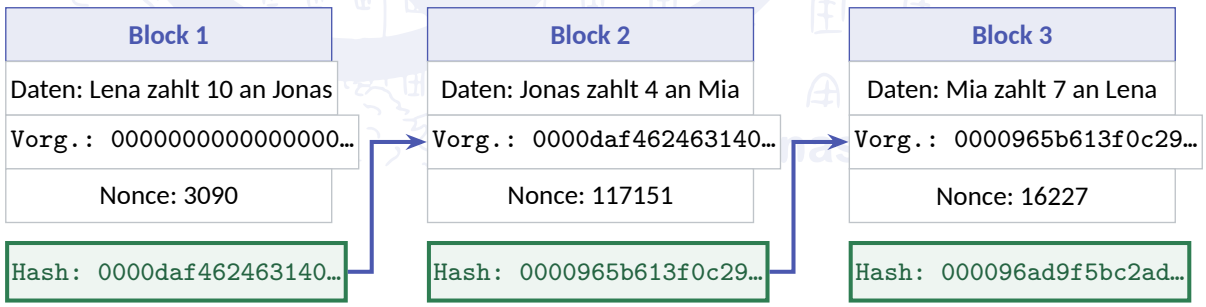
Leitidee: Die Frage hinter der Blockchain

Kann man ein Kassenbuch führen, das **alle** besitzen, in das **jeder** schreiben darf — und in dem trotzdem niemand nachträglich etwas ändern kann?

Die erste Hälfte ist leicht: Man schickt das Buch an alle. Die zweite ist das Problem. Wer eine Kopie hat, kann darin radieren.

8.12.2 Die Kette

Der Trick ist eine Kette aus Hashwerten. Die Zeilen werden zu **Blöcken** zusammengefasst, und jeder Block enthält — neben seinen eigenen Daten — den **Hashwert seines Vorgängers**.



Jeder Hash deckt alles ab, was über ihm im Block steht — auch den Hash des Vorgängers.

Eine Blockchain mit drei Blöcken. Die Werte sind echte SHA-256-Hashes.

Was das bewirkt, sieht man am besten beim Versuch zu betrügen. Angenommen, jemand ändert in Block 1 den Betrag von 10 auf 100. Dann ändert sich der Hashwert von Block 1 vollständig:

```
vorher:  0000daf462463140…
nachher: 3448594b61837fab…
```

Block 2 trägt aber noch den alten Wert als Vorgänger-Hash. Die Kette passt nicht mehr zusammen, und zwar sichtbar für jeden, der nachrechnet. Wer die Fälschung verbergen will, muss auch Block 2 neu berechnen — und dann Block 3, und alle weiteren.

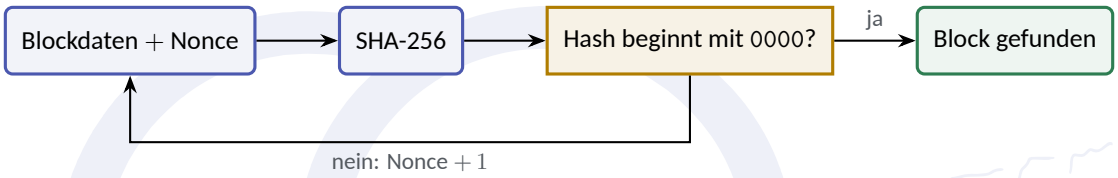
Merke

Eine Blockchain macht Änderungen nicht *unmöglich*. Sie macht sie **auffällig**: Wer einen alten Block ändert, muss die gesamte Kette dahinter neu bauen.

8.12.3 Mining: Arbeit als Bremse

Neu bauen wäre für einen Rechner eine Sache von Millisekunden. Deshalb kommt eine zweite Zutat hinzu: Ein Block gilt nur dann als gültig, wenn sein Hashwert eine **Bedingung** erfüllt — zum Beispiel, dass er mit mehreren Nullen beginnt.

Am Hashwert selbst lässt sich nicht drehen; er ergibt sich aus dem Inhalt. Also baut man ein Feld in den Block ein, das sonst keinen Zweck hat: die **Nonce** (von *number used once*). Man zählt sie hoch und rechnet jedes Mal den Hash neu, bis die Bedingung zufällig erfüllt ist. Dieses Suchen heißt **Mining**.



Mining ist Ausprobieren. Einen anderen Weg gibt es nicht.

Beispiel: Wie teuer die Bedingung ist

Gehasht wird jeweils die Zeichenkette aus Datensatz, einem senkrechten Strich und der Nonce, also `Lena zahlt 10 an Jonas|0`, dann `Lena zahlt 10 an Jonas|1` und so fort. Gesucht ist die kleinste Nonce, bei der der SHA-256-Wert mit einer bestimmten Zahl von Nullen beginnt:

Bedingung	gefundene Nonce	Versuche	Rechenzeit
`0…`	7	8	unmessbar
`00…`	370	371	unmessbar
`000…`	9 815	9 816	0,01 s
`0000…`	10 021	10 022	0,01 s
`00000…`	4 728 918	4 728 919	3,4 s

Jede zusätzliche Null macht die Suche im Mittel **sechzehnmal** teurer — eine Hexadezimal stelle hat sechzehn mögliche Werte. Das echte Bitcoin-Netz arbeitet derzeit mit einer Bedingung, für die ungefähr 10^{23} Versuche nötig sind.

Merke

Prüfen ist billig, Finden ist teuer. Diese Asymmetrie ist der ganze Mechanismus: Ein einziger Hashdurchlauf bestätigt, was Billionen Versuche gekostet hat. Man nennt das **Proof of Work** — Arbeitsnachweis.

8.12.4 Was eine Kryptowährung daraus macht

Eine Kryptowährung ist dieses Kassenbuch, verteilt auf viele tausend Rechner.

- Wer eine Überweisung tätigen will, **signiert** sie mit seinem privaten Schlüssel (Abschnitt 8.11). Die „Kontonummer" ist der zugehörige öffentliche Schlüssel. Es gibt kein Konto, keinen Namen, keine Bank — nur ein Schlüsselpaar.
- Die Rechner im Netz sammeln Überweisungen und suchen nach einer gültigen Nonce. Wer zuerst eine findet, verbreitet seinen Block, und alle anderen hängen ihn an.
- Als Lohn darf der Finder sich selbst eine festgelegte Menge neuer Münzen gutschreiben. Daher der Name *Mining*: Es sieht aus, als würde Geld geschürft, tatsächlich wird eine Rechenleistung bezahlt.
- Gehen zwei Ketten auseinander, gilt die Regel: **Die längste Kette zählt.** Wer eine Fälschung durchsetzen will, muss also schneller rechnen als der gesamte Rest des Netzes zusammen.

Leitidee: Was wirklich gelöst wurde

Das Problem war nie die Verschlüsselung — die gab es längst. Gelöst wurde die Frage, wie sich Fremde, die einander nicht trauen und keine gemeinsame Instanz anerkennen, auf **eine** Reihenfolge von Ereignissen einigen. Der Preis dafür ist, dass diese Einigung Rechenarbeit kostet, und zwar absichtlich.

8.12.5 Was es kostet

Die absichtliche Verschwendung ist kein Nebeneffekt, sondern das Sicherheitsmerkmal — und sie hat eine Größenordnung. Das Bitcoin-Netz verbrauchte zuletzt Strom in der Größenordnung eines mittleren europäischen Staates, für eine Zahl von Überweisungen, die ein einzelner Bankrechner nebenbei abwickelt. Neuere Netze wie Ethereum sind deshalb auf **Proof of Stake** umgestiegen: Dort entscheidet nicht Rechenarbeit, wer den nächsten Block anhängen darf, sondern ein hinterlegtes Pfand.

Dazu kommen Eigenschaften, die je nach Standpunkt Vorzug oder Mangel sind:

Eigenschaft	Folge
keine zentrale Stelle	niemand kann Konten sperren — und niemand hilft bei Verlust
Schlüssel statt Konto	wer den privaten Schlüssel verliert, verliert alles, endgültig
unumkehrbare Buchungen	kein Rückruf bei Irrtum oder Betrug
öffentliche Kette	jede Überweisung ist für immer einsehbar, pseudonym, nicht anonym
Kurs ohne Deckung	der Wert folgt allein Angebot und Nachfrage

Merke

Die Technik ist nachprüfbar, die Bewertung ist es nicht. Dieser Exkurs erklärt, **wie** eine Blockchain funktioniert — ob sie für eine bestimmte Aufgabe das richtige Werkzeug ist, ist eine wirtschaftliche und politische Frage, keine informatische.

Arbeitsauftrag 12

46. Berechnen Sie mit einem Programm die SHA-256-Werte von `Informatik` und `informatik`. Wie viele Stellen stimmen überein? Welche Eigenschaft einer Hashfunktion zeigt sich daran?
47. Schreiben Sie ein Programm, das zu einem Text die kleinste Nonce sucht, deren SHA-256-Wert mit `000` beginnt. Gehasht wird `<Text>|<Nonce>` wie in der Tabelle oben; in Python genügt `import hashlib`.
48. Erweitern Sie es auf vier und fünf Nullen und messen Sie die Zeit. Vergleichen Sie mit der Tabelle oben. Warum weichen Ihre Nonce-Werte nicht ab, Ihre Zeiten aber schon?
49. Bauen Sie eine Kette aus drei Blöcken nach. Ändern Sie dann den Inhalt von Block 1 und zeigen Sie an den Hashwerten, dass die Kette bricht.
50. Begründen Sie, warum ein Angreifer mit 10 % der Rechenleistung des Netzes eine Fälschung praktisch nicht durchsetzen kann.
51. Erläutern Sie den Unterschied zwischen *pseudonym* und *anonym* am Beispiel einer öffentlichen Blockchain.

Selbstcheck

Können Sie in zwei Sätzen erklären, warum eine Blockchain ohne Proof of Work nichts nützt — und was genau die Nonce mit Sicherheit zu tun hat?

Gymnasium Ottweiler

8.13 Zusammenfassung

- Verschlüsselung schützt die Vertraulichkeit; Integrität und Authentizität brauchen Hashfunktionen und Signaturen.
- Nach Kerckhoffs darf nur der Schlüssel geheim sein, nie das Verfahren.
- Klassische Verfahren kombinieren Transposition und Substitution. Monoalphabetische Substitution fällt der Häufigkeitsanalyse zum Opfer, polyalphabetische dem Kasiski-Test.
- Ein großer Schlüsselraum schützt nur vor Brute Force.
- Das One-Time-Pad ist beweisbar sicher, aber praktisch kaum einsetzbar.
- Symmetrische Verfahren (AES) sind schnell, haben aber das Schlüsselverteilungsproblem.
- Der Schlüsseltausch von Diffie und Hellman löst es ohne jedes Geheimnis auf der Leitung: Beide Seiten potenzieren modular, und das Potenzgesetz sorgt dafür, dass dasselbe herauskommt. Rückwärts steht der diskrete Logarithmus im Weg.
- Asymmetrische Verfahren (RSA) gehen einen Schritt weiter und lösen es über Einweg- und Falltürfunktionen.
- Kein Schlüsselaustausch sichert, *mit wem* man spricht — das ist die Lücke, in die sich ein Man-in-the-Middle setzt und die erst Zertifikate schließen.
- Die Praxis kombiniert beides: asymmetrischer Schlüsselaustausch, symmetrische Datenverschlüsselung, Signaturen und Zertifikate für die Echtheit.

Wiederholungsaufgaben

52. Erläutern Sie an einem selbst gewählten Beispiel den Unterschied zwischen einem passiven und einem aktiven Angriff.
53. Vergleichen Sie Cäsar-, Vigenère- und One-Time-Pad-Chiffre in einer Tabelle nach Schlüssellänge, Schlüsselraum und bekanntem Angriff.
54. Ein Mitschüler behauptet: „Ein Verfahren mit 10^{26} Schlüsseln kann niemand knacken.“ Nehmen Sie begründet Stellung.
55. Skizzieren Sie den Ablauf einer HTTPS-Verbindung und markieren Sie, an welcher Stelle symmetrisch und an welcher asymmetrisch gearbeitet wird.
56. Erklären Sie, warum RSA mit $n = 91$ zwar rechenbar, aber vollkommen unsicher ist.

Gymnasium Ottweiler