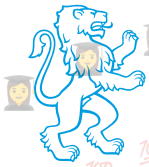


Kantonsschule Im Lee

Informatik

Kryptologie

Skript

Hauptautor
Cyril Wendl

Ko-Autor
Thomas Graf

Inhaltsverzeichnis

1 Kryptologie **2**
1.1 Einführung . . . 2

2 Symmetrische Kryptosysteme **5**
2.1 Verschlüsselung per Transposition . . . 5
2.1.1 Skytale . . . 5
2.2 Caesar . . . 6
2.2.1 Knacken von Caesar . . . 7
2.3 Monoalphabetische Substitution . . . 9
2.4 Vigenère . . . 10
2.4.1 Knacken von Vigenère . . . 13
2.4.1.1 Bestimmung der Schlüssellänge mit dem Kasiski-Test . . . 16
2.4.1.2 Bestimmung der Schlüssellänge: Friedman'sche Charakteristik . . . 19
2.5 One-Time-Pad . . . 24
2.5.1 Kryptoanalyse bei mehrfacher Verwendung des Schlüssels . . . 27
2.5.2 Bin-One-Time-Pad (OTP) . . . 28

3 Schlüsseltausch-Verfahren **31**
3.1 Drei-Wege-Schlüsseltausch . . . 32
3.2 Diffie-Hellman-Merkle-Schlüsseltausch . . . 34

4 Asymmetrische Kryptosysteme **36**
4.1 RSA-Verfahren . . . 37
4.1.1 Digitale Signaturen mit RSA . . . 40

A Python-Übungen zu Kryptologie **42**
A.1 Allgemeine Zeichenketten-Aufgaben . . . 42
A.2 Verschlüsselung von Texten in Python . . . 43
A.3 Caesar-Verschlüsselung in Python . . . 48
A.4 Vigenère-Verschlüsselung in Python . . . 50

B Lernziele Kryptologie **53**

Kapitel 1

Kryptologie

1.1 Einführung

Alice möchte Bob eine wichtige Nachricht zukommen lassen, beispielsweise zu einem gesundheitlichen Problem oder um ihre Bankkontoverbindung mit Bob zu teilen. Da das Internet jedoch eine *offene* und somit (grundsätzlich) unsichere Technologie ist, kann jedermann jede Nachricht mitlesen. Somit könnte eine bösartige Person, wie beispielsweise Eve, die Nachricht abhören, um sich Zugriff auf die Gesundheitsdaten oder das Bankkonto von Alice zu verschaffen.

Damit dies nicht passieren kann, muss Alice ihre Nachricht so verschlüsseln, dass nur Bob sie lesen kann. Dieses Problem hat sich bereits in der Antike (und vermutlich noch vorher gestellt) und entspricht einem grundsätzlichen, menschlichen Bedürfnis: Wie kann ein Feldherr seinen Soldaten Anweisungen geben, ohne dass der Gegner mithört? Oder, um eine andere Situation aufzugreifen: Wie können Sie sich mit Ihren Geschwistern austauschen, ohne dass Ihre Eltern verstehen, worum es geht?

In der Informatik spricht man hier davon, dass Alice aus ihrem „Klartext“, also dem für alle Menschen verständlichen Text, einen „Kryptotext“ macht, also einen Text, den nur Bob entschlüsseln kann, d.h., nur Bob weiss, wie man aus diesem Text wieder einen verständlichen Text macht. Hierzu muss man sich auf eine Verschlüsselungsmethode einigen. In der Informatik spricht man hierbei von einem **Verschlüsselungs-Algorithmus** und **Entschlüsselungs-Algorithmus**. Die Methode, um eine Nachricht zu verschlüsseln, so dass sie für Dritte unlesbar ist, wird häufig auch „**Schlüssel**“ genannt, und das Verfahren, um eine Nachricht unlesbar zu machen „Verschlüsselung“ oder „Chiffrierung“. Im Allgemeinen wird der Bereich der Informatik, der sich mit Ver- und Entschlüsselung befasst, „Kryptografie“ genannt und die verschiedenen Algorithmen und Ansätze werden häufig als „Kryptosysteme“ bezeichnet.

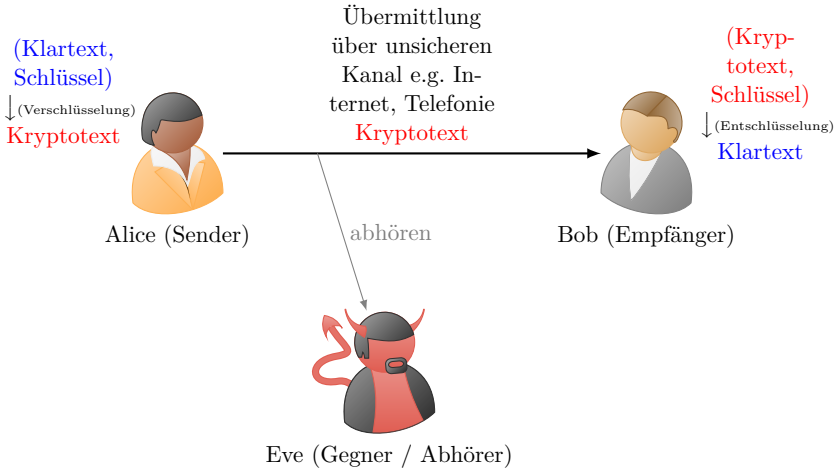


Abbildung 1.1: Dieses Schema zeigt die Kommunikation zwischen Alice (Sender) und Bob (Empfänger) unter Verwendung eines Kryptosystems.

Im Folgenden setzen wir uns zuerst mit einigen grundlegenden Verschlüsselungs-Methoden auseinander, um zu verstehen, was ein sicheres Kryptosystem auszeichnet. Der niederländische Kryptologe und Linguist Auguste Kerckhoffs stellte im Jahr 1883 sechs Grundsätze für sichere Verschlüsselungsverfahren auf:

Abbildung 1.2: Auguste Kerckhoffs (1835-1903)

1. Das System muss **unentzifferbar** sein.
2. Das System darf **keiner Geheimhaltung** bedürfen.
3. Das System muss **leicht übermittelbar** sein und man muss sich die **Schlüssel ohne schriftliche Aufzeichnung merken** können.
4. Das System sollte **mit telegraphischer Kommunikation kompatibel** sein.
5. Das System muss **transportabel** sein und die Bedienung darf nicht mehr als eine Person erfordern.
6. Das System muss **einfach anwendbar** sein.

Ein System, das diese Anforderungen erfüllt, gab es damals nicht. Von besonderer Wichtigkeit war seine Forderung nach Öffentlichkeit des Kryptosystems:

> „Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi." – Auguste Kerckhoffs, *La cryptographie militaire* (1883)

Demgegenüber steht die Auffassung, dass Kryptosysteme geheimgehalten werden sollten (*Security through Obscurity*), eine Haltung, die häufiger von militärischen Institutionen sowie kommerziellen Anbietern von Verschlüsselungsmethoden verfechtet wird.

Folgende Sicherheits-bezogenen Anforderungen können zusätzlich an moderne Kryptosysteme gestellt werden:

1. **Vertraulichkeit**: Es soll sichergestellt sein, dass wirklich nur diejenige Person eine Nachricht lesen kann, für die diese bestimmt ist.
2. **Integrität**: Der Empfänger soll feststellen können, ob die Nachricht nach ihrer Erzeugung verändert wurde (wir wollen ja die originale Nachricht!).
3. **Authentizität**: Die Verfasserin einer Nachricht soll identifizierbar sein, bzw. der Empfänger soll nachprüfen können, wer die Verfasserin ist.
4. **Verbindlichkeit**: Die Verfasserin soll nicht abstreiten können, dass sie die Verfasserin der Nachricht ist.

Kapitel 2

Symmetrische Kryptosysteme

2.1 Verschlüsselung per Transposition

In einem mit Transposition (oder Permutation) verschlüsselten Text bleiben die Buchstaben des Klartexts im Kryptotext erhalten, ändern aber die Reihenfolge.

Aufgabe 2.1

Entziffern Sie den folgenden Kryptotext durch ausprobieren:

1. RKPYOTOLIGEEMREOLGCITHEGEHMIINSSE
2. HCSFIRILTAHCZFUEBUHAWNERDNUKUZMMOINUEIZNER

2.1.1 Skytale

Das Kyrptosystem *Skytale* wurde bereits von den Griechen für militärische Zwecke verwendet. Der Verschlüsselungsalgorithmus kann wie folgt beschrieben werden:

- Schreibe den Klartext zeilenweise in eine Tabelle (Matrix) von links nach rechts.
- Allfällige leere Felder in der letzten Zeile der Tabelle werden mit beliebigen Buchstaben gefüllt.
- Den Kryptotext erhalten wir, indem wir die Buchstaben Spalte für Spalte von links nach rechts und von oben nach unten lesen.

Praktisch umgesetzt werden kann dieser Algorithmus mit einem Stab und einem Band, siehe Abbildung 2.2. Die Anzahl Zeichen, die auf eine Windung des Bandes um den Stab passen, entspricht der Anzahl Zeilen in der Tabelle. Diese Anzahl, also die Anzahl Zeilen, entspricht dem Schlüssel des Skytale-Verschlüsselungsverfahrens.

Abbildung 2.2: Praktische Umsetzung der Skytale-Verschlüsselung mit einem Stab und einem Band
Quelle

Aufgabe 2.2

Der Kryptotext

WNGIAEIEMATSMKRTTEAGIEINANINTUGNDOJEEENL

wurde mit einer Tabelle mit 5 Zeilen und 8 Spalten erzeugt. Wie lautet der Klartext?

Aufgabe 2.3

Der folgende Kryptotext der Länge 75 wurde mit SKYTALE verschlüsselt:

ETIFIITNUTNFGENKURRELEODIERSILIMSEANIE
MRECREMRHSNSSAPSTCBRCRHEUHORRNHNIEGEG

Dabei konnten mit dem Klartext **alle Zeilen der Tabelle vollständig aufgefüllt** werden.

1. Welches ist hier der Schlüssel, d.h. die Anzahl Zeichen auf einer Windung (bzw. Anzahl Zeilen der Tabelle)? Tipp: Probieren Sie die Schlüssel 3 und 5 aus. Sie können diesen Link, Entschlüsselungswerkzeug „Tabelle“ dazu verwenden.
2. Wie lautet der Klartext?
3. Wie viele Schlüssel müssen im schlimmsten Fall ausprobiert werden, bis der korrekte Schlüssel gefunden wurde? Das heisst, wie viele potentielle Schlüssel gibt es? Es gilt immer noch, dass alle Zeilen der Tabelle vollständig befüllt waren.

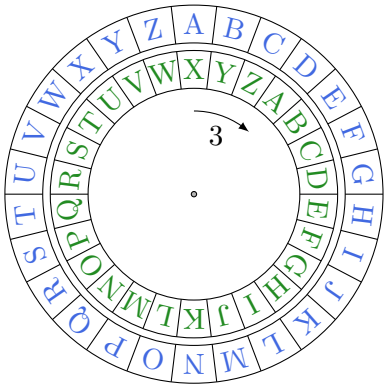
Aufgabe (Challenge) 2.4

Sie möchten einen Klartext der Länge 87 mit Hilfe einer Tabelle mit 10 Zeilen chiffrieren. Wie viele Spalten benötigt diese Tabelle?

2.2 Caesar

Eines der bekanntesten Verschlüsselungssysteme der Antike ist die Caesar-Verschlüsselung, die von Julius Caesar verwendet wurde. Dieses Verschlüsselungssystem besteht essentiell aus einer Verschiebung aller Buchstaben um eine vordefinierte Anzahl Positionen im Alphabet. Der ***Schlüssel*** bezeichnet dabei die Anzahl Positionen, um die jeder Buchstabe verschoben wird. Ausgedrückt wird der Schlüssel auch als Buchstabe, der dem Buchstaben A entsprechen würde.

Beispiel 2.1:
Mit dem Schlüssel „D“ (3 Positionen) würde das Wort „HALLO“ als „KDOOR“ geschrieben:



Der Schlüssel „B“ bedeutet also eine Verschiebung um 1 Position, „C“ um 2 Positionen, „D“ um drei Positionen usw. Im Allgemeinen lässt sich die Verschiebung folgendermassen schreiben:

Verschiebung = Ord(□), wobei Ord(A) = 0.

Beispiel 2.2:
Folgender Text wurde mit dem Schlüssel „G“ (6 Positionen) verschlüsselt:

```
  Klartext: JEMANDMUSSTEJOSEFKVERLEUMDETHA...
 Schlüssel: GGGGGGGGGGGGGGGGGGGGGGGGGGGGG...
Kryptotext: PKSGTJSAYYZKPUYKLQBKXRKASJKZNG...
```

Aufgabe 2.5

Der Kryptotext

`X G T Y G P F G U E J N W G U U G N B Y G K`

wurde mit CAESAR verschlüsselt, der Schlüssel ist aber unbekannt. Entschlüsseln Sie den Kryptotext, ohne alle Schlüssel auszuprobieren, wenn Sie wissen, dass der häufigste Buchstabe im Klartext `E` ist.

2.2.1 Knacken von Caesar

- Einerseits reicht es, alle 25 möglichen Verschiebungen auszuprobieren. Ein moderner Computer hat dies in einigen Milisekunden erledigt.
- Andererseits kann, wenn der Text genügend lange ist, anhand der Häufigkeit der verschlüsselten Buchstaben mittels Häufigeitsanalyse in kürzester Zeit bestimmt werden, was der Schlüssel war. Abbildung 2.3 zeigt die Häufigkeiten der Buchstaben eines langen, mit Caesar verschlüsselten Texts im Kryptotext.

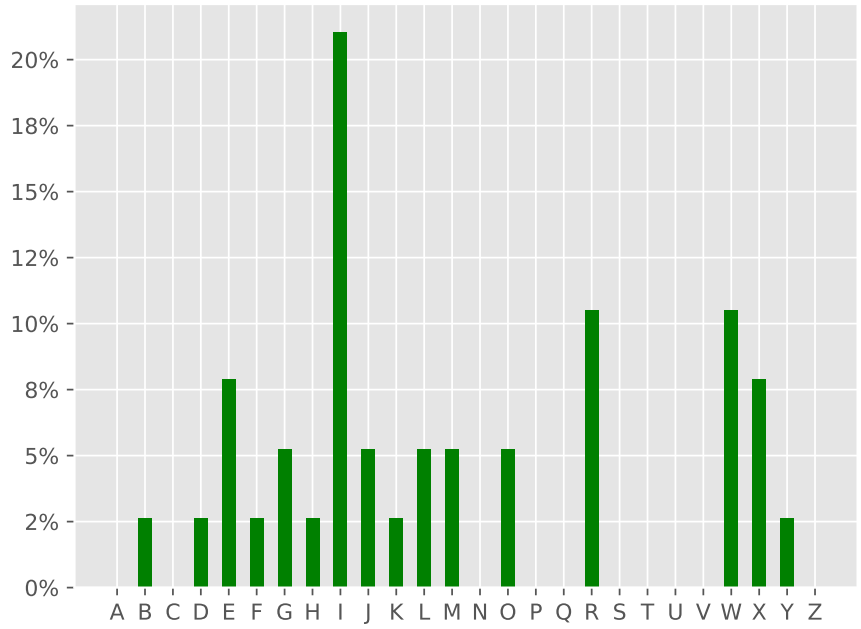


Abbildung 2.3: Buchstabenhäufigkeit in einem nach Caesar verschlüsselten Text

Die durchschnittlichen Häufigkeiten von Buchstaben, Bi- und Tri-Grammen in deutschen Texten ist in Tabelle 2.1 angegeben.

Buchstabe	Relative Häufigkeit (%)
E	17.40
N	9.78
I	7.55
S	7.27
R	7.00
A	6.51
T	6.15
D	5.08
H	4.76
U	4.35

(a) Buchstabenhäufigkeit

Bigramm	Relative Häufigkeit (%)
ER	3.94
EN	3.07
CH	2.73
DE	2.41
EI	2.29
ND	2.07
IE	1.97
GE	1.88
TE	1.88
IN	1.82

(b) Bigrammhäufigkeit

Trigramm	Relative Häufigkeit (%)
DER	1.44
SCH	1.21
ICH	1.08
DIE	0.98
UND	0.95
DEN	0.78
CHE	0.77
EIN	0.75
NDE	0.74
GEN	0.72

(c) Trigrammhäufigkeit

Tabelle 2.1: Relative Häufigkeiten der Buchstaben, Bigramme und Trigramme im Deutschen

Aufgabe 2.6

Können Sie aus Tabelle 2.1 entschlüsseln, was der Klartext ist?

Kryptotext:

`WMILEFIRIWKIWGLEJJXHMIWIRXIBXDYOREGOIR`

Aufgabe (Challenge) 2.7

Kann man immer wie in Aufgabe 2.6 vorgehen? In welchen Fällen funktioniert dieses Vorgehen eventuell nicht?

Aufgabe (Challenge) 2.8

Um zu verschlüsseln, liest man die Caesar-Scheibe von innen (Klartext) nach aussen (Kryptotext). Um zu entschlüsseln, liest man von aussen (Kryptotext) nach innen (Klartext). Gibt es Schlüssel bei Caesar, bei denen es sowohl für die Ver- wie Entschlüsselung keine Rolle spielt, in welche Richtung man die Scheiben liest?

2.3 Monoalphabetische Substitution

Eine Weiterentwicklung der Caesar-Veschlüsselung besteht darin, nicht jeden Buchstaben im Klartext um dieselbe Anzahl Positionen zu verschieben, sondern für jeden Klartext-Buchstaben B_K einen Kryptotext-Buchstaben B_G zu definieren, durch den der Buchstabe ersetzt wird (siehe Abbildung 2.4).

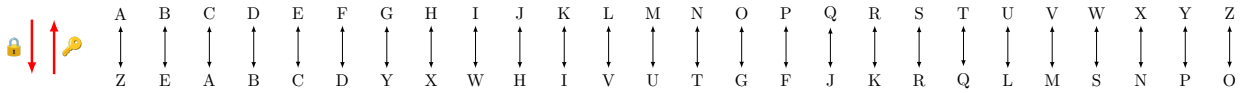


Abbildung 2.4: Monoalphabetische Substitution

Diese Verschlüsselungsmethode kann nicht geknackt werden, indem alle Buchstaben um gleich viele Positionen verschoben werden. Man kann also nicht lediglich alle 25 möglichen Verschiebungen ausprobieren.

Obschon es nun eine Vielzahl möglicher Schlüssel gibt, kann diese Verschlüsselungsmethode ebenfalls sehr einfach geknackt werden per Häufigkeitsanalyse, die wir bereits in Abschnitt 2.2 gesehen und in Aufgabe 2.6 mittels Tabelle 2.1 durchgeführt haben.

Beispiel 2.3:
Folgender Kryptotext wurde abgefangen:

`MUMMUXJUQYMHQOUSUTUQNGWTJQVHUMXQUXQJSUVYPPUM`

Wir wissen, dass er per monoalphabetische Verschlüsselung erstellt wurde, wie beispielsweise in Abbildung 2.4, kennen jedoch den Schlüssel nicht. Wir verwenden Tabelle 2.1, um den Schlüssel zu erraten und den Text zu entschlüsseln. Dabei gehen wir wie folgt vor:

- Wir beginnen damit, die häufigsten Buchstaben zu zählen: U kommt 10-mal vor, M kommt 6-mal vor, Q kommt 6-mal vor. Aufgrund von Tabelle 2.1 versuchen wir, folgendes einzusetzen: `U=E, M=N, Q=I`. Wir erhalten folgenden Teil-Klartext:
  ```
  KRYPTOTEXT: MUMMUXJUQYMHQOUSUTUQNGWTJQVHUMXQUXQJSUVYPPUM
  KLARTEXT:   NENNE--EI-N-I-E-E-EI-----I--EN-IE-I--E----EN
  ```

- Nun könnten wir weiterfahren, indem wir die häufigsten Trigramme anschauen. Dabei suchen wir im bisherigen Klartext nach Klartext-Teilen, wo wir Trigramme einsetzen könnten. Laut Tabelle 2.1 ist ein häufiges Trigramm in deutschen Texten ``DIE''. Dieses probieren wir einzusetzen (D=X):
  ```
  KRYPTOTEXT: MUMMUXJUQYMHQOUSUTUQNGWTJQVHUMXQUXQJSUVYPPUM
  KLARTEXT:   NENNED-EI-N-I-E-E-EI-----I--ENDIEDI--E----EN
  ```

- Der Klartext nimmt langsam Form an und wir können nun damit beginnen, verbleibende Worte zu erraten. Könnte es sich beim ersten Wort um das Wort "DREI" handeln? Wir setzen ein:
  ```
  KRYPTOTEXT: MUMMUXJUQYMHQOUSUTUQNGWTJQVHUMXQUXQJSUVYPPUM
  KLARTEXT:   NENNEDREI-N-I-E-E-EI----RI--ENDIEDIR-E----EN
  ```

- Sobald einzelne Wörter erkannt werden, können wir sie mit Trennlinien voneinander abgrenzen:
  ```
  KRYPTOTEXT: MUMMU|XJUQ|YMHQOUSUTUQNGWTJQVHUM|XQU|XQJ|SUVYPPUM
  KLARTEXT:   NENNE|DREI|-N-I-E-E-EI----RI--EN|DIE|DIR|-E----EN
  ```
- Wir fahren durch Ausprobieren und Erraten weiter, bis das Lösungswort gefunden ist:
  ```
  KRYPTOTEXT: MUMMU|XJUQ|YMHQOU|SUTUQNGWTJQVHUM|XQU|XQJ|SUVYPPUM
  KLARTEXT:   NENNE|DREI|ANTIKE|GEHEIMSCHRIFTEN|DIE|DIR|GEFALLEN
  ```

Je länger ein Text ist, desto zuverlässiger funktioniert das Erraten der Buchstaben per Häufigkeitsanalyse.

Aufgabe 2.9

Entschlüsseln Sie den Text "ECRRCKZVRAZCRZK" mit dem Schlüssel aus Abbildung 2.4.

Aufgabe (Challenge) 2.10

Lösen Sie eine Knobelaufgabe aus dem Kapitel "Substitution".

Aufgabe (Challenge) 2.11

Wie viele Verschlüsselungs-Möglichkeiten gibt es in der Verschlüsselungsmethode aus Abbildung 2.4?

2.4 Vigenère

Wie wir in Aufgabe 2.6 gesehen haben, ist es extrem einfach, Texte, die mit Caesar verschlüsselt worden sind, anzugreifen, entweder mittels Buchstabenhäufigkeitsanalyse oder indem man einfach alle 25 möglichen Verschiebungen ausprobiert. Auch weitere monoalphabetische Substitutions-Verfahren

wie Abbildung 2.4 können durch etwas Knobeln relativ einfach geknackt werden.

Vigenère hatte eine andere Idee: Statt den ganzen Text mit einem einzigen Schlüssel zu verschlüsseln, verwendete er ein Wort, mit welchem er den Text "zyklisch", also gruppenweise verschlüsselte.

Beispiel 2.4:
Wenn der Schlüssel beispielsweise "KEY" war, wurden die Buchstaben folgendermassen verschlüsselt (siehe Beispiel unterhalb):

- Buchstaben 1, 4, 7, 10 etc. mit "K"
- Buchstaben 2, 5, 8, 11 etc. mit "E"
- Buchstaben 3, 6, 9, 12 etc. mit "Y"

```
  Klartext: JEM|AND|MUS|STE|JOS|EFK|VER|LE...
 Schlüssel: KEY|KEY|KEY|KEY|KEY|KEY|KEY|KE...
Kryptotext: TIK|KRB|WYQ|CXC|TSQ|OJI|FIP|VI...
```

Im Vergleich dazu wird bei Caesar jeder Buchstabe durch *denselben* Schlüssel verschlüsselt, beispielsweise:

```
  Klartext: JEMANDMUSSTEJOSEFKVERLEUMDETHA...
 Schlüssel: GGGGGGGGGGGGGGGGGGGGGGGGGGGGG...
Kryptotext: PKSGTJSAYYZKPUYKLQBKXRKASJKZNG...
```

Wie Sie wissen, ist der Buchstabe "E" der häufigste Buchstabe in deutschen Texten. Diese Eigenschaft haben wir uns in Aufgabe 2.6 zunutze gemacht, um den Text zu knacken. Da bei Vigenère jedoch der Buchstaben nun nicht mehr immer mit demselben Schlüssel verschlüsselt ist, sondern je nach Position mit dem Schlüssel "K", "E", oder "Y", wird der Buchstabe "E" im Kryptotext nun breiter auf andere Buchstaben verteilt (siehe Abbildung 2.5). Anders gesagt, ein Buchstabe im Kryptotext repräsentiert nun nicht mehr immer den gleichen Buchstaben im Klartext, sondern kann einen von drei Buchstaben repräsentieren.

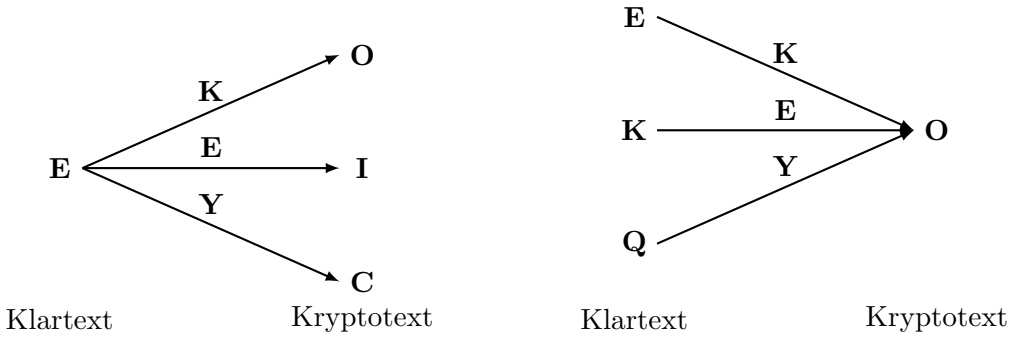
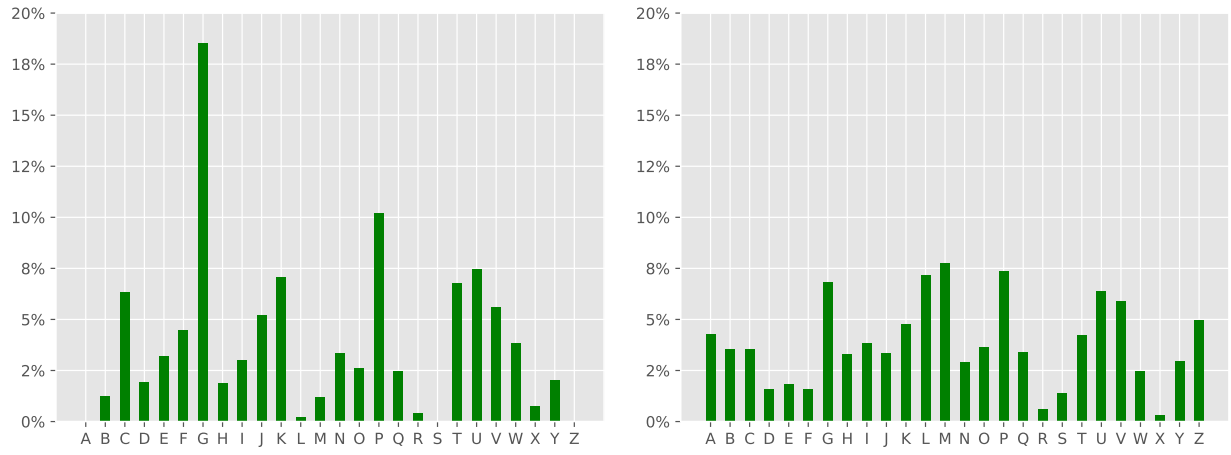


Abbildung 2.5: Verschlüsselungs-Möglichkeiten mit Vigenère, mit dem Schlüssel "KEY" aus Beispiel 2.4

Diese Verschlüsslungsmethode galt während mehreren Jahrehunderten als sicher und war der Gold-Standard in vielen militärischen Verschlüsslungs-Anwendungen. Um einen Text zu verschlüsseln, wurde jeder Buchstabe im Klartext einzeln verschlüsselt, und je nach Schlüssel (z.B. "K", "E" oder "Y" in Beispiel 2.4) wurde der entsprechende Buchstabe des Kryptotexts aus einer Tabelle herausgelesen (siehe Tabelle 2.2)

Abbildung 2.6 zeigt die Buchstabenhäufigkeit im Kryptotext für einen langen Text, welcher einmal mit Caesar und einmal mit Vigenère verschlüsselt wurde.



(a) Caesar

(b) Vigenère

Abbildung 2.6: Buchstabenhäufigkeit in einem langen Text, welcher mit unterschiedlichen Verfahren verschlüsselt wurde

Abbildung 2.6 zeigt, dass Vigenère zu einer insgesamt homogeneren Verteilung aller Buchstaben im Kryptotext führt. Dies kommt daher, dass bei Vigenère jeder Klartext-Buchstaben auf *mehrere* Kryptotext-Buchstaben verteilt wird, währenddem bei Caesar jeder Klartext-Buchstaben durch genau *einen* Buchstaben im Kryptotext kodiert wird. Tabelle 2.2 zeigt die Vigenère-Tabelle, mithilfe welcher man einen Text mit der Vigenère-Methode ver- sowie entschlüsseln kann, sofern der Schlüssel bekannt ist. Folgende zwei Beispiele zeigen auf, wie diese Tabelle genutzt werden kann:

1. **Verschlüsselung**: Soll beispielsweise der Klartext-Buchstabe „E“ mit dem Schlüsselteil „K“ verschlüsselt werden, so findet man in der Tabelle den entsprechenden Kryptotext-Buchstaben „O“.
2. **Entschlüsselung**: Soll beispielsweise der Kryptotext-Buchstabe „O“ mit dem Schlüsselteil „K“ entschlüsselt werden, so sucht man in der Tabelle nach dem Klartext-Buchstaben, der mit „O“ und „K“ korrespondiert. Dies ist in diesem Fall der Buchstabe „E“.

Klartextbuchstabe \ Schlüsselbuchstabe	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

Tabelle 2.2: Vigenère-Tabelle

Aufgabe 2.12

Entschlüsseln Sie folgenden Text von Hand, wenn Sie wissen, dass er mit dem Schlüssel „TOP“ und mit der Methode von Vigenère verschlüsselt worden ist:

`UFPOCVNHVXAPVVI`

Aufgabe 2.13

Verschlüsseln Sie folgenden Text von Hand, mit dem Schlüssel „YES“ und mit der Methode von Vigenère:

`NIEMANDKANNDASLESEN`

2.4.1 Knacken von Vigenère

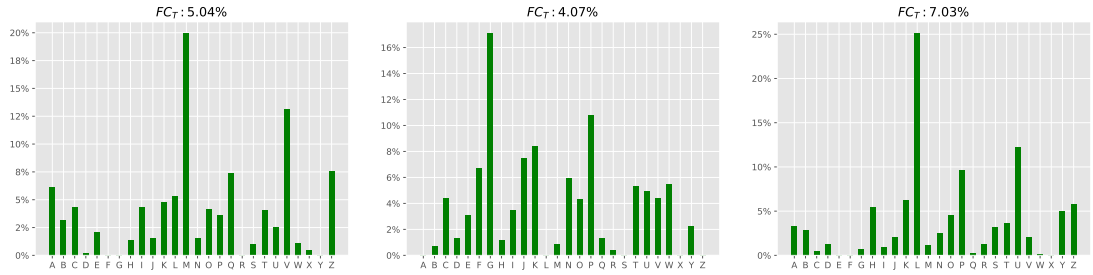
Wenn die Schlüssellänge eines mit Vigenère verschlüsselten Texts bekannt ist, ist dieser relative einfach zu knacken: Man unterteilt den Text in diesem Fall einfach in Gruppen und geht für jede

Gruppe gleich vor wie bei Caesar, um den Schlüssel zu finden.

Beispiel 2.5:
Angenommen, wir wüssten, dass folgender Text mit einem Schlüssel verschlüsselt worden ist, der drei Zeichen lang ist:

MKU MYB VFL ZDH ZGO MKA MTR MKA PCA UGP VGN IPG MUL MNL MKU OGU WOT MPN TGP KJK MPZ CGZ
AGU NTB MJS QPN AOV ZIL VFP MKJ POP BIH VBL UJL ZBL VIL VKL AUL QEO JKU INS MKU CPK NTL
CGT QEO UGP VGZ TGI MPZ QPK QGZ MTN MIL VFK QGM CGY AQS KJL AGL TGU OGZ KJH NHL VKZ BYP
MFP MOL QPL QEO JKU AQN TWL KMS QEO UGP VDL AVL ZUV OCU HKU LGT OGM CGO TGC WPY CJP OGT
LCZ MKU DGY AWU SGU LCZ AOL QPL SWU AVK ITB VVL ZNL QFL BKJ PMV MPU BGQ MVG BPP KJA HGP
KJU MPU QEO BGP VGU AVY QEO CPK JKU VKL MKU OTV MUZ MTL ZOH TGY OGD MUL VCS AKU LKL AGU
IWN MPI TKJ SGU EGU VFH ANP MDL

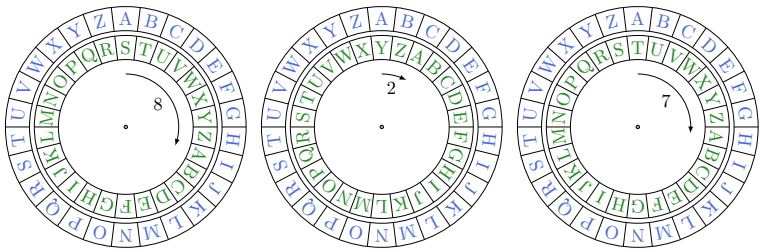
Den Schlüssel selbst kennen wir nicht, wir wissen jedoch, dass jeder der grün markierten Buchstaben beispielsweise mit demselben Buchstaben verschlüsselt worden ist. Wir können also für jede der Gruppen (grün, blau, rot) die Häufigkeit der Buchstaben betrachten (siehe Abbildung 2.7).



(a) Untergruppe 1 (b) Untergruppe 2 (c) Untergruppe 3

Abbildung 2.7: Buchstabenhäufigkeit in einem mit Vigenère verschlüsselten Text, pro Untergruppe

Dabei erkennen wir, dass die häufigsten Buchstaben pro Gruppe M, G, bzw. L sind. Wir haben es also mit folgenden Verschiebungen zu tun:



Der Schlüssel muss demnach “ICH” sein und wir können den Text somit entschlüsseln.

Aufgabe 2.14

Kopieren Sie den folgenden Kryptotext und versuchen Sie, ihn mit dem Analysetool zu entziffern, wenn Sie wissen, dass die Schlüssellänge drei ist. Schauen Sie sich die Buchstabenhäufigkeiten für jede der 3 Gruppen an, indem Sie unter „Analysewerkzeug“ die Häufigkeitsanalyse auswählen und *stride* (Schrittgrösse) 3. Unterhalb der Grafiken kann man zwischen jeder der drei Gruppen hin- und herwechseln. Probieren Sie den wahrscheinlichsten Schlüssel aus, indem Sie beim „Entschlüsselungswerkzeug“ „Vigenère“ auswählen und den Schlüssel eingeben.

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

Aufgabe 2.15 Vigenère knacken bei bekannter Schlüssellänge (zu dritt)

Finden Sie für folgenden Vigenère-Text den Klartext heraus, wenn Sie wissen, dass der Schlüssel Länge 3 hat:

```
WKJ LMU AKJ WZJ WQV WVV KBG ZBG JAV OMP FRG VMT WZM WVP LEK WMN WUG FBC JMK FMX WZN
SMU KTK UPG NMT KKJ DCG KAG DCP YNW WZW FAG JMH JMK ZMK LQU L
```

1. Finden Sie zuerst die häufigsten Buchstaben pro Gruppe heraus (Gruppe 3 ist bereits gemacht, s. unten). Seien Sie genau, ansonsten müssen Sie später wieder von vorne beginnen!

   Notizen

2. Finden Sie danach den Schlüssel heraus, indem Sie annehmen, dass der häufigste Buchstabe im Geheimtext den Buchstaben „`E`“ im Klartext repräsentiert. Für Gruppe 3 ist die Lösung bereits vorgegeben.

	Häufigster Buchstabe	**Schlüssel**
Gruppe 1		
Gruppe 2		
Gruppe 3	`(E →) G`	`(A →) C` (=2 Verschiebungen)

3. Enschlüsseln Sie nun den Geheimtext, indem Sie die Caesar-Drehscheibe verwenden (Tipp: Machen Sie jeweils eine Farbgruppe pro Mal).

```
--- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- --- ---
--- --- --- --- --- --- --- --- --- --- --- --- --- --- --- -
```

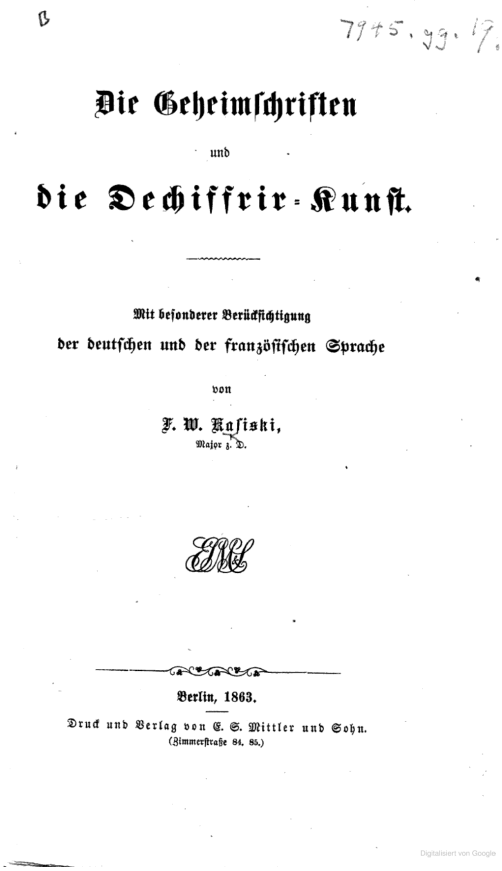
Aufgabe (Challenge) 2.16

Versuchen Sie, auf diesem Link eine weitere Challenge-Aufgabe zu lösen, indem Sie die Werkzeuge „Frequency“ (Häufigkeit) und „Vigenère“ verwenden:

2.4.1.1 Bestimmung der Schlüssellänge mit dem Kasiski-Test

Wie Sie gesehen haben, kann man Vigenère *gruppenweise* mit denselben Methoden knacken, die wir benutzt haben, um Caesar zu knacken. Falls die Schlüssellänge unbekannt ist, können sowohl der Kasiski-Test wie auch die Friedman'sche Charakteristik verwendet werden, um diese zu erraten. In diesem Unterabschnitt schauen wir uns zunächst den Kasiski-Test an.

Friedrich Kasiski war ein preussischer Infanteriemajor (1805 - 1868), welcher mit seinem Kasiski-Test massgeblich zum Knacken der Vigenère-Verschlüsselung beigetragen hat. Den Kasiski-Test veröffentlichte er 1863 in seinem Buch „Die Geheimschriften und die Dechiffrir-Kunst“ (siehe Abbildung 2.8).



B 7945. 99. 17.

Die Geheimschriften

und

die Dechiffrir-Kunst.

Mit besonderer Berücksichtigung
der deutschen und der französischen Sprache

von

F. W. Kasiski,
Major z. D.

Berlin, 1863.
Druck und Verlag von E. S. Mittler und Sohn.
(Zimmerstraße 94, 95.)

Digitalisiert von Google

Abbildung 2.8: Umschlag des Buches „Die Geheimschriften und die Dechiffrir-Kunst“ von Friedrich Kasiski (1863, Quelle)

Die Grundidee hinter dem Kasiski-Test ist: Wenn ein Wort oder ein Wortteil im Klartext mehrmals vorkommen, so wird dieser möglicherweise auch im Kryptotext mehrmals vorkommen. Wenn beispielsweise die Buchstaben „DIE“ mehrmals mit dem Schlüssel „KEY“ verschlüsselt werden, dann wird der Text „LIV“ mehrmals im Geheimtext auftauchen. Wenn das Wort „DIE“ also zweimal im Klartext vorkommt **und** auf denselben Schlüsselteil („KEY“, „EYK“ oder „YKE“) fällt, wird z.B. „LIV“ auch zweimal im Kryptotext gleich verschlüsselt sein.

Wenn wir also im Kryptotext nach wiederholten Sequenzen suchen, können wir daraus die Schlüssellänge ableiten.

Beispiel 2.6:
In folgendem Beispiel ist der Klartext auf der ersten Zeile. Der Schlüssel “CODE” wird verwendet, um den Gemeimtext (auf der dritten Zeile) zu erzeugen:

```
MEINKLEINERREIMSCHEINTKEINERZUSEIN

CODECODECODECODECODECODECODECODECO

OSLRMZHMPSUVGWPWEVHMPHNIKBHVBIVIKB
  2     7           19     24       32
```

Der Kasiski-Test erlaubt uns, Hinweise auf die Schlüssellänge zu erhalten, ohne den Schlüssel zu kennen:

- Die Sequenzen „HMP“ und „IKB“ erscheinen jeweils zweimal im Geheimtext.
- Die Sequenz HMP befindet sich an den Positionen 7 und 19. Die Differenz zwischen diesen Positionen beträgt 12, was sich als $2 \times 2 \times 3$ in Primfaktoren zerlegen lässt.

  $$\underset{\text{Position 1 HMP}}{\mathbf{7}} + \underset{\text{Codewort-Länge!}}{\mathbf{4}} \times \underset{\text{Anzahl Codewörter}}{\mathbf{3}} = \underset{\text{Position 2 HMP}}{\mathbf{19}}$$

- Die Sequenz IKB befindet sich an den Positionen 24 und 32. Die Differenz zwischen diesen Positionen beträgt 8, was sich als $2 \times 2 \times 2$ faktorisieren lässt.

  $$\underset{\text{Position 1 IKB}}{\mathbf{24}} + \underset{\text{Codewort-Länge!}}{\mathbf{4}} \times \underset{\text{Anzahl Codewörter}}{\mathbf{2}} = \underset{\text{Position 2 IKB}}{\mathbf{32}}$$

- Die Schlüssellänge könnte, sofern man nur den Abstand zwischen den beiden HMP betrachtet, theoretisch auch 12 sein. Dann würden allerdings die beiden IKB nicht auf denselben Schlüsselteil fallen. Daher ist dies nicht möglich.
- Den Primfaktoren $2 \times 2 \times 3$ und $2 \times 2 \times 2$ ist der Teil 2×2 gemeinsam, was darauf hindeutet, dass die **Schlüssellänge 4** sein könnte.

Aufgabe 2.17 Vigenère knacken bei bekannter Schlüssellänge (zu dritt)

Finden Sie den Klartext für folgende Nachricht heraus, wenn Sie wissen, dass der Text mit Vigenère verschlüsselt worden ist. Wenden Sie den Kasiski-Test an!

```
UEUIOOCEUDIWIOOWRRCLWVUQURRCLWVNDTHXETHERRCFKRTWVSFYOQRNJJTG
          10        20        30        40        50        60

RSVUAVIOOCEQEIHRUIYOHITQLNJZNJVSEVRJRUILNGUEUIOOCEUDIWIOOWHR
          70        80        90       100       110       120

VRWVAXWZXIOOCEQIOOWAWDEWVAXWUQUWRCLWVDLVNDVCKJTHETDXEYFMUFLO
          130       140       150       160       170       180

VRQZCKKSPVHUYOHIEQ
          190
```

Anleitung:

- Die Position an allen 10, 20, 30 etc. Buchstaben sind oben bereits markiert, ebenso wie Vorkommnisse des Trigramms IOO.
- Auf separatem Blatt:
  1. Notieren Sie sich die **Anfangspositionen** aller Trigramme IOO.
  2. Berechnen Sie die **Abstände** zwischen den Anfangspositionen. Es reicht, wenn Sie nur die Abstände aller Anfangspositionen von IOO zum ersten Trigramm IOO an Position 4 berechnen.
  3. Zerlegen Sie die Abstände zwischen den Trigrammen in **Primfaktoren**.
  4. Finden Sie den **grössten gemeinsamen Teiler** dieser Primfaktoren, um die Schlüssellänge zu erraten.
- Wenn Sie die Schlüssellänge haben, färben Sie den Text abwechslungsweise ein, wie in Aufgabe 2.15. Teilen Sie sich zu dritt auf, so dass jede Person eine Gruppe von Buchstaben entschlüsselt (gleich wie in Aufgabe 2.15).
- **Tipp**: Der letzte Teil des Schlüssels ist "D". Der Schlüssel ergibt ein Wort. Entschlüsseln Sie den Text nun mit der Caesar-Drehscheibe! Entschlüsseln Sie zuerst nur die erste Zeile, um zu überprüfen, ob ihr Schlüssel stimmt und zu einem sinnvollen Text führt.

Aufgabe 2.18

Finden Sie die Schlüssellänge für folgenden Vigenère-Text heraus, indem Sie den Kasiski-Test anwenden:

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

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

Verwenden Sie dazu folgendes Online-Tool: https://cryptbreaker.marcwidmer.xyz/solve

1. Verwenden Sie das Analyse-Werkzeug „Kasiski-Analyse“, um die Abstände der sich wiederholenden Sequenzen der Länge 4 zu bestimmen.
   - Notieren Sie sich die Positionen und Abstände der sich wiederholenden Sequenzen der Länge 4.
   - Zerlegen Sie die Abstände in Primfaktoren.
   - Finden Sie den grössten gemeinsamen Teiler der Abstände, um die Schlüssellänge zu erraten.
2. Überprüfen Sie Ihre Vermutung, indem Sie im Analyse-Werkzeug die Häufigkeitsanalyse mit der Schrittweite (*stride*) gleich der vermuteten Schlüssellänge durchführen.
3. Entschlüsseln Sie den Text mit dem „Vigenère“-Entschlüsslungswerkzeug.

Aufgabe 2.19

1. Erklären Sie kurz: Dopplungen im Klartext (z.B. „EIN“) führen nicht unbedingt zu Dopplungen im Geheimtext.
2. Erklären Sie kurz: Dopplungen im Geheimtext müssen nicht in jedem Fall aus Dopplungen im Klartext stammen; sie können auch zufällig entstehen.

Aufgabe (Challenge) 2.20

Lösen Sie den Kasiski-Test online:

2.4.1.2 Bestimmung der Schlüssellänge: Friedman’sche Charakteristik

Eine weitere Möglichkeit, die Schlüssellänge zu erraten, besteht in der Verwendung der sogenannten Friedman’schen Charakteristik.

William Friedman (1891-1969) war ein russisch-amerikanischer Kryptologe und Pionier auf dem Gebiet der Kryptoanalyse (siehe Abbildung 2.9). Kurz vor dem Ausbruch des zweiten Weltkriegs gründete er den Signals Intelligence Service (SIS), eine Geheimabteilung des US-Militärs zur Entzifferung feindlicher Codes und Geheimschriften.

Abbildung 2.9: William Friedman (Quelle)

Die Friedman'sche Charakteristik macht sich zunutze, dass Buchstaben im Klartext ungleich verteilt sind. Falls jeder Buchstabe gleich häufig vorkommen würde, wäre die erwartete Häufigkeit jedes Buchstabens genau 1/26. In keiner natürlichen Sprache besitzen alle Buchstaben dieselbe relative Häufigkeit. Die Häufigkeitsverteilung der Buchstaben in deutschen Texten haben wir bereits in Tabelle 2.1 gesehen. Die relative Häufigkeit des Buchstabens „E" in deutschen Texten ist beispielsweise ca. 17.4%. Die relative Häufigkeit eines Buchstabens □ in einem Text T bezeichnen wir mit $h_{\square}(T)$.

Die Friedman'sche Charakteristik berechnet, wie ungleich alle Buchstaben in einem Text verteilt sind, indem sie die quadrierte Differenz der relativen Häufigkeit jedes Buchstabens zu 1/26 berechnet. Diese quadrierten Abweichungen werden danach alle aufsummiert (siehe. Gleichung (2.1)).

Bemerkung 2.1:
Das Quadrieren der Abweichungen (Differenzen) hat zwei Effekte:

- Quadrate von reellen Zahlen sind sicherlich nicht negativ.
- Grosse Abweichungen von 1/26 tragen aufgrund des Quadrierens überproportional zur Summe (zur Friedmansch'schen Charakteristik) bei.

$$FC(T) = \left(h_A(T) - \frac{1}{26}\right)^2 + \left(h_B(T) - \frac{1}{26}\right)^2 + \ldots + \left(h_Z(T) - \frac{1}{26}\right)^2 \tag{2.1}$$

$$= \sum_{\Delta \in \text{Alphabet}} \left(h_\Delta(T) - \frac{1}{26}\right) \tag{2.2}$$

Eine stark ungleiche Verteilung von Buchstaben in einem Text T führt also zu einer hohen Friedman'schen Charakteristik $FC(T)$, währenddem gleichmässig verteiltere Buchstaben im Text T zu einer tieferen $FC(T)$ führen. Deutschsprachige (Klar-)Texte haben durchschnittlich etwa einen Wert von 3.8% (0.038) (siehe Abbildung 2.10).

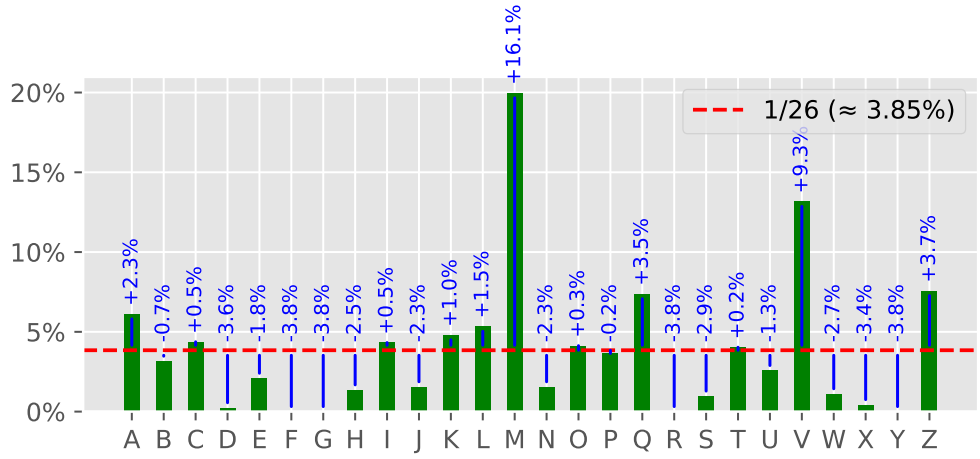
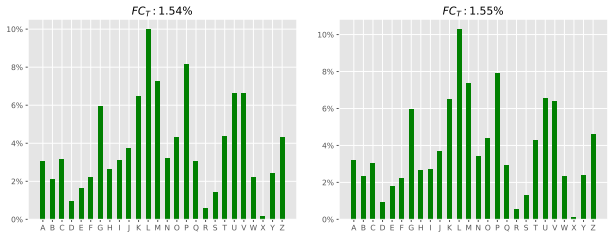


Abbildung 2.10: Buchstabenhäufigkeit in einem mit Caesar verschlüsselten Text, im Vergleich zur Gleichverteilung von Buchstaben

Je grösser die blauen Pfeile in Abbildung 2.10, desto höher die Friedman'sche Charakteristik.

Wir können nun folgenden *Brute-Force*-Ansatz verwenden, um die Schlüssellänge mit der Friedman'schen Charakteristik zu bestimmen:

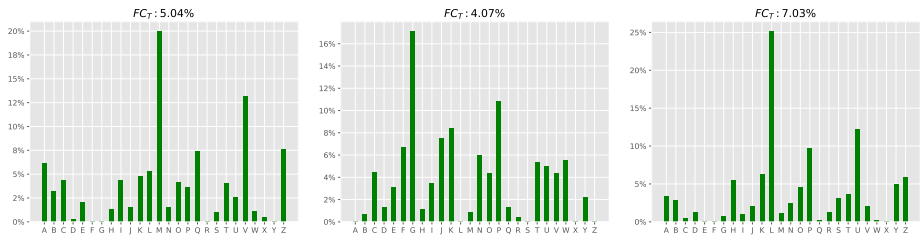
- Alle möglichen Schlüsselwortlängen von 1 bis zu einer beliebig gewählten Zahl n ausprobieren, wobei n praktisch nie über 10 gewählt wird.
- Buchstaben jeweils in Gruppen unterteilen:
  - **2 Gruppen**: MU ZK JL QP AW JU MH YI IL LC ZA UP MR LZ HL SV CM TZ BC UL GU PC IM PD QG TI PC QI LV GY MG UB UJ PN BM UZ MN AP GY HN PK JL OT HB WS IV PW PK IH B



(a) Gruppe 1 (b) Gruppe 2

Abbildung 2.11: Buchstaben-Häufigkeiten und FC_T für Schlüsssellänge=2

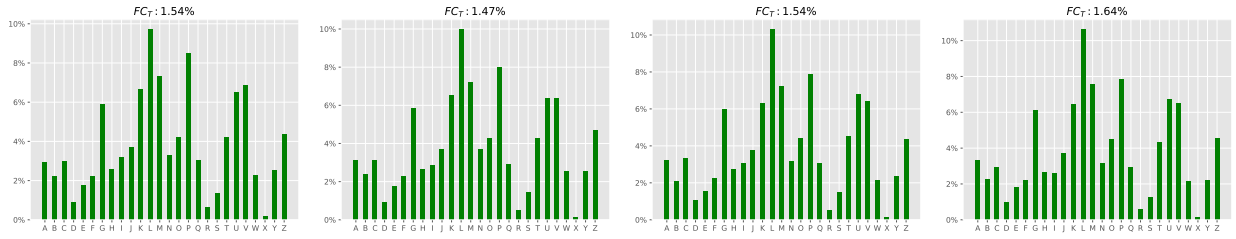
  - **3 Gruppen**: MUZ KJL QPA WJU MHY IIL LCZ AUP MRL ZHL SVC MTZ BCU LGU PCI MPD QGT IPC QIL VGY MGU BUJ PNB MUZ MNA PGY HNP KJL OTH BWS IVP WPK IHB



(a) Gruppe 1 (b) Gruppe 2 (c) Gruppe 3

Abbildung 2.12: Buchstaben-Häufigkeiten und FC_T für Schlüsssellänge=3

  - **4 Gruppen**: MUZK JLQP AWJU MHYI ILLC ZAUP MRLZ HLSV CMTZ BCUL GUPC IMPD QGTI PCQI LVGY MGUB UJPN BMUZ MNAP GYHN PKJL OTHB WSIV PWPK IHB



(a) Gruppe 1 (b) Gruppe 2 (c) Gruppe 3 (d) Gruppe 4

Abbildung 2.13: Buchstaben-Häufigkeiten und FC_T für Schlüsssellänge=4

  - etc, bis zu Schlüssellänge = n
- Für jede der Schlüssellängen: Durchschnittliche $FC(T)$ für alle Gruppen berechnen.
- Wenn die richtige Schlüssellänge (oder ein Vielfaches davon) gewählt wurde, sollte $FC(T)$ höher sein.

Die durschschnittliche Friedman'sche Charakteristik für jede der getesteten Schlüssellängen ist gezeigt in Abbildung 2.14.

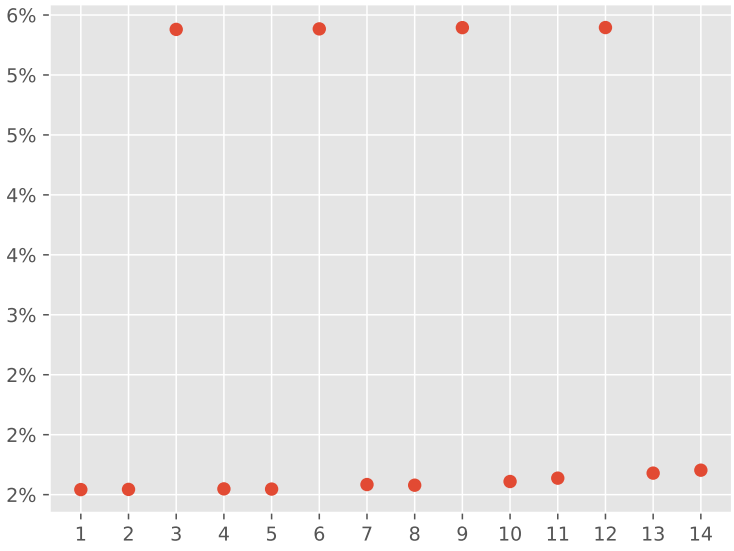


Abbildung 2.14: Durchschnittliche Friedman'sche Charakteristik für jede Schlüssellänge

Aus Abbildung 2.14 lässt sich ablesen, dass die Schlüssellänge vermutlich 3 sein muss, da bei jedem Vielfachen der Schlüssellänge 3 die durchschnittliche Friedman'sche Charakteristik höher ist als für die restlichen Schlüssellängen. Ab nun lässt sich gleich vorgehen wie in Unterabschnitt 2.4.1 beschrieben, um den Text zu entschlüsseln.

Aufgabe 2.21

Berechnen Sie die Friedman'sche Charakteristik für folgende beide Texte von Hand:

P A P P E R L A P A P P

B A C K S T E I N

Wie interpretieren Sie die beiden Zahlen?

Aufgabe 2.22

Für welche Buchstabenverteilung sollte die Friedman'sche Charakteristik höher sein: Abbildung 2.6, (a) oder (b)?

Aufgabe 2.23

Kopieren Sie den folgenden Kryptotext und versuchen Sie, ihn mit dem Analysetool zu entziffern, indem Sie zuerst das Tool "Friedman" und danach das Tool "Frequency" verwenden. Wie lautet der Schlüssel?

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

```
H V S X V B W F Z B J J T R W F U I Z A F Z N Z W D Y B D B T F G G I F T K G W D Y M Z W O S E N H F
I S J U
```

Aufgabe (Challenge) 2.24

Berechnen Sie die Friedman'sche Charakteristik für folgenden Kryptotext:

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

Verwenden Sie dazu das Python-Skript auf Moodle, mit der Funktion `def calculate_fc(text)`. Was entnehmen Sie der ihrer Antwort? Wurde der Text mit Caesar oder mit Vigenère verschlüsselt?

Aufgabe (Challenge) 2.25 Ver- und Entschlüsslung mit Python (zu zweit)

Laden Sie zuerst die Vigenère-Python-Dateien von Moodle herunter. Erstellen Sie danach einen Text in deutscher Sprache mit mindestens 1000 Zeichen, beispielsweise mittels folgender Webseite: https://www.blindtextgenerator.de/. Diesen werden Sie nun mit Python verschlüsseln.

Teil 1: Verschlüsselung

- Verschlüsseln Sie Ihren Text, indem Sie die Funktion `def vigenere(text, key, encrypt=True)` verwenden.
- Senden Sie Ihren verschlüsselten Text an Ihre(n) Partner(in), nicht aber den Schlüssel.

Teil 2: Entschlüsselung

- Sie erhalten den Kryptotext und müssen nun zuerst den Schlüssel herausfinden. Bestimmen Sie diese mithilfe der Friedman'schen Charakteristik, indem Sie die Funktion `def get_friedman_vals(text, maxkeylen)` verwenden.
- Nachdem Sie die Schlüssellänge bestimmt haben, finden Sie innerhalb jeder Gruppe dem häufigsten Buchstaben. Dies können Sie mit der Funktion `def show_letter_freq(text)` einfach umsetzen. Somit sollten Sie das Schlüsselwort herausfinden können.
- Entschlüsseln Sie nun den Kryptotext, indem Sie die Funktion `def vigenere(text, key, encrypt=False)` verwenden.

Aufgabe (Challenge) 2.26

Lösen Sie drei "beliebige Probleme" auf der Analyse-Webseite.

2.5 One-Time-Pad

Der Fortschritt in der Mathematik und das neu dazugekommene Wissen machten Vigenère zu einem unsicheren Kryptosystem.

Vigenère und dessen Kryptoanalyse haben wir bereits besprochen. Wir haben gesehen, dass eine

Kryptoanalyse vor allem dann leicht ist, wenn der verwendete Schlüssel zu kurz gewählt wird, denn die wesentliche Schwäche von Vigenère ist die Wiederholung der Muster im Kryptotext bei zu kurz gewähltem Schlüssel. Betrachten wir als Beispiel einen Kryptotext aus 1000 Buchstaben, der mit einem Schlüssel der Länge fünf verschlüsselt ist. Das bedeutet, dass jeder fünfte Buchstabe und somit insgesamt je 200 Buchstaben anhand der gleichen Zeile der Vigenère-Tabelle verschlüsselt sind. Das heisst, dass je 200 Buchstaben mit dem gleichen Schlüsselbuchstaben verschlüsselt sind. Durch eine Häufigkeitsanalyse von 200 Buchstaben kann ein Kryptoanalytiker bereits den entsprechenden Buchstaben des Schlüssels bestimmen. Was wäre aber, wenn der verwendete Schlüssel aus 50 Buchstaben bestehen würde? Dann muss eine Häufigkeitsanalyse von 50 Teilen zu je 20 Buchstaben gemacht werden. Es ist nicht garantiert, dass man aus nur 20 Buchstaben eine repräsentative Häufigkeitsverteilung erhält. Gehen wir noch einen Schritt weiter und wählen einen Schlüssel, der genau gleich lang ist wie der Klartext. Nun ist eine Häufigkeitsanalyse völlig unmöglich, da wir es mit 1000 Teilen zu je nur einem Buchstaben zu tun haben.

Die One-Time-Pad-Verschlüsselungsmethode (OTP, deutsch “Einmalschlüssel-Verfahren”) funktioniert im Prinzip identisch wie die Vigenère-Methode, mit folgenden drei Unterschieden:

1. Der Schlüssel besteht auf einer *zufälligen* Folge von Buchstaben
2. Der Schlüssel ist *genau gleich lang* wie der Klartext/Kryptotext
3. Der Schlüssel wird nur für genau eine Botschaft verwendet

Beispiel 2.7:
Folgendes Beispiel verwendet einen OTP-Schlüssel:

```
  Klartext: OERLIKON
 Schlüssel: IGBQPWXD
Kryptotext: WKSBXGLQ
```

Essentiell handelt es sich beim OTP um dasselbe Verschlüsselungsverfahren wie bei Vigenère, wobei der Schlüssel gleich lang wie die zu verschlüsselnde Nachricht sein muss. Diese Methode gilt als sicher, da die gruppenweise Häufigkeitsanalyse (z.B. mit der Friedman’schen Charakteristik) nicht funktioniert. Allerdings ist die OTP-Methode aufgrund der längeren Schlüssellänge mit höheren Übertragungskosten verbunden. Zudem darf jeder Schlüssel zur Sicherheit nur einmal verwendet werden, was für regelmässige Datenaustausch-Anwendungen wie E-Mail, Online-Banking etc. unpraktisch ist.

Vorgehen 2.1 (Kryptosystem OTP)**:**

Klartextalphabet: Alphabet der lateinischen Grossbuchstaben.
Kryptotextalphabet: Alphabet der lateinischen Grossbuchstaben.
Schlüsselmenge: Alle denkbaren Texte bestehend aus lateinischen Grossbuchstaben, welche dieselbe Länge haben wir der Klartext. Es ist wichtig, dass für jeden Klartext ein Schlüssel zufällig generiert wird.
Verschlüsselung: Gegeben ist ein zufällig gewählter Schlüssel s aus der Schlüsselmenge. Der gegebene Klartext wird nun (wie gewohnt) mit Hilfe der Vigenère-Tabelle mit dem Schlüssel s verschlüsselt. Der Schlüssel darf danach nicht mehr verwendet werden.
Entschlüsselung: Gegeben ist ein zufällig gewählter Schlüssel s aus der Schlüsselmenge. Der gegebene Kryptotext wird (wie gewohnt) durch Vigenère mit dem Schlüssel s entschlüsselt.

Warum erscheint uns das OTP als ein sicheres Kryptosystem? Die Intuition ist wie folgt. Weil der

Schlüssel zufällig gewählt wird und genauso lang ist wie der Klartext, wird jeder Buchstabe des Klartextes um zufällig viele Positionen im Alphabet verschoben. Damit kann man den Kryptotext als eine zufällige Folge von Buchstaben betrachten. Und aus einer zufälligen Folge von Buchstaben kann man keine Informationen herauslesen.

Alice und Bob haben sich in einem geheimen treffen schon vor einigen Tagen auf den geheimen Schlüssel der Länge 5 für das OTP geeinigt. Alice verwendet nun den mit Bob vereinbarten Schlüssel, um eine geheime Nachricht (Kryptotext) an ihn zu senden. Der gesendete Kryptotext lautet `GVRCL`.

Eve hat den Nachrichtenaustausch belauscht und somit den Kryptotext in Erfahrung gebracht. Sie möchte nun den Klartext herausfinden um zu erfahren, was Alice und Bob unternehmen werden. Eve vermutet, dass der Kryptotext mit dem sicheren OTP verschlüsselt ist. Da der verwendete Schlüssel gleich lang ist wie der Klartext, ist eine Kryptoanalyse mit der Häufigkeitsanalyse unmöglich.

Aufgabe 2.27

(a) Wie viele mögliche Schlüssel der Länge 5 gibt es?
(b) Kann Eve den Kryptotext `GVRCL` entschlüsseln, falls sie (im schlimmsten Fall) alle Möglichkeiten durchprobiert?

Dennoch hat Eve das Gefühl, dass sie die geheime Nachricht erraten kann. Die Anzahl aller Klartexte, die aus fünf Buchstaben einen sinnvollen Text ergeben, wird vermutlich nicht so gross sein. Ausserdem weiss Eve, dass sich Alice und Bob verabreden wollen. Eve listet deshalb einige sinnvolle Texte zu je fünf Buchstaben auf. Für jeden dieser möglichen Klartexte bestimmt sie den Schlüssel (mit Hilfe der Vigenère-Tabelle), der den entsprechenden Text zu dem gegebenen Kryptotext `GVRCL` verschlüsseln würde.

möglicher Klartext	entsprechender Schlüssel
BADEN	FVOYY
ESSEN	CDZYY
LESEN	VRZYY
SPORT	OGDLS
VIDEO	LNOYX

Tabelle 2.3: Entsprechende Schlüssel bei geratenen möglichen Klartexten (BADEN, ESSEN, LESEN, SPORT, VIDEO) für abgehörten Kryptotext GVRCL.

Jeder dieser Texte kann also durch den angegebenen Schlüssel zum Kryptotext `GVRCL` verschlüsselt werden. Welcher Text entspricht nun der richtigen Nachricht? Alice und Bob haben ihren geheimen Schlüssel zufällig bestimmt, das heisst, jeder mögliche Schlüssel kann mit der gleichen Wahrscheinlichkeit ausgewählt werden. Eve hat daher keine Möglichkeit herauszufinden, welche dieser vier möglichen Klartexte der geheimen Nachricht entspricht. Es ist auch möglich, dass der richtige Klartext nicht in der Liste steht. Somit hat Eve keine Chance irgendeinen Teil des Klartextes oder des Schlüssels zu erfahren.

Ein Hauptproblem aller symmetrischen Verschlüsselungsverfahren besteht jedoch darin, dass der Schlüssel erst einmal über einen *sicheren* Kanal ausgetauscht werden muss. Vor dem Internet erfolgte dies durch einen Postboten, heute ist dies allerdings nicht mehr praktikabel. Dies werden wir im nächsten Kapitel besprechen.

Aufgabe 2.28

Wie viele mögliche Schlüssel hat der OTP für eine Nachricht der Länge n?

2.5.1 Kryptoanalyse bei mehrfacher Verwendung des Schlüssels

Nun wollen wir wissen, weshalb ein Schlüssel beim OTP nur einmal verwendet werden darf. Dazu schauen wir uns einen erneuten Nachrichtenaustausch zwischen Alice und Bob an. Alice möchte Bob nämlich eine weitere geheime Nachricht schicken. Da die zwei jedoch zuvor keinen zweiten Schlüssel vereinbart haben, verwendet Alice den gleichen Schlüssel ein zweites Mal. Diesmal erhält Bob von Alice den folgenden Kryptotext: `ADRCM`.

Eve hat ihr Vorhaben, die beiden zu belauschen, noch nicht aufgegeben und versucht erneut die verschlüsselte Mitteilung zu lesen. Wenn Alice und Bob für die zweite Nachricht einen neuen zufälligen Schlüssel ausgemacht hätten, dann könnte Eve erneut nichts mit dem Kryptotext anfangen. Alice war jedoch nachlässig und verwendete den gleichen Schlüssel ein zweites Mal, um sich mit Bob zu verabreden. Eve ergänzt ihre Tabelle mit einer dritten Spalte. In dieser Spalte notiert sie den Klartext, der entsteht, wenn sie den zweiten Kryptotext mit dem entsprechenden Schlüssel aus der zweiten Spalte entschlüsselt.

möglicher Klartext	entsprechender Schlüssel	möglicher Klartext für die zweite Nachricht
BADEN	FVOYY	VIDEO
ESSEN	CDZYY	YASEO
LESEN	VRZYY	FMSEO
SPORT	OGDLS	FMSEO
VIDEO	LNOYX	PQDEP

Tabelle 2.4: Entschlüsslung eines weiteren abgehörten Kryptotextes (ADRCM) durch die vorher bestimmten denkbaren Schlüsselkandiaten. Der Schlüsselkandidat FVOYY erzeugt aus beiden abgehörten Kryptotexten einen sinnvollen Klartext.

Und siehe da, fast alle Texte in der dritten Spalte ergeben keinen Sinn, ausser dem Text in der ersten Zeile. Eve erkennt, dass mit dem Schlüssel FVOYY sowohl der erste wie auch der zweite Kryptotext zu einem sinnvollen Text entschlüsselt werden können. Durch den Vergleich der Entschlüsselungen des ersten und des zweiten Kryptotextes bei gleichem Schlüssel konnte Eve die tatsächlichen Klartexte und den Schlüssel herausfinden.

2.5.2 Bin-OTP

Das Kryptosystem Bin-OTP funktioniert fast gleich wie das OTP, nur dass wir nicht mit dem lateinischen Alphabet der Grossbuchstaben arbeiten wollen, sondern lediglich mit dem binären Alphabet $\{0, 1\}$. Aus der grossen Vigenère-Tabelle in Tabelle 2.2 wird im binären Alphabet die übersichtliche (binäre) Vigenère-Tabelle:

Klartextbuchstabe \ Schlüsselbuchstabe	0	1
0	0	1
1	1	0

Abbildung 2.15: Vigenère-Tabelle für das binäre Alphabet.

Der Tabelle entnehmen wir, dass der Klartextbuchstabe 0 durch den Schlüssel 0 zum Kryptotextbuchstaben 0 wird. Man verwendet für die Verschlüsslung mit der binären Vigenère-Tabelle eine besondere Schreibweise. Für die Verschlüsslung des Klartextbuchstaben 0 durch den Schlüssel 0 schreiben wir

$$0 \oplus 0 = 0.$$

Wird 0 durch 1 verschlüsselt erhalten wir den Kryptotextbuchstaben 1 (siehe Tabelle) und schreiben

$$0 \oplus 1 = 1.$$

Bitte beachten Sie, dass auch

$$1 \oplus 0 = 1$$

sowie

$$1 \oplus 1 = 0$$

gilt. Die Verschlüsslung mit dem binären OTP erfolgt nun, indem Bit für Bit die Operation $\oplus$ (gemäss binärer Vigenère-Tabelle) durchgeführt wird. Analog haben wir mit den lateinischen Buchstaben auch die Verschlüsselung Buchstabe für Buchstabe mithilfe der Vigenère-Tabelle durchgeführt.

Beispiel 2.8:
Folgendes Beispiel illustriert, wie die Verschlüsselung mit dem Bin-OTP funktioniert:

- Der Klartext ist gegeben durch 101 und der Schlüssel durch 111. Dann ist der Kryptotext gegeben durch $101 \oplus 111 = 010$:

$$\begin{array}{rccc} & 1 & 0 & 1 \\ \oplus & 1 & 1 & 1 \\ \hline & 0 & 1 & 0 \end{array}$$

- Der Klartext ist gegeben durch 011101 und der Schlüssel durch 110001. Dann ist der Kryptotext gegeben durch

$$011101 \oplus 110001 = 101100.$$

Aufgabe 2.29

Berechnen Sie den Kryptotext zu dem gegebenen Klartext

$$110010100$$

und zum Schlüssel

$$101110001.$$

Aufgabe 2.30

(a) Berechnen Sie sowohl

$$100110 \oplus 001011$$

als auch

$$001011 \oplus 100110.$$

Was stellen Sie fest?

(b) Seien a und b zwei beliebige Bits. Begründen Sie, warum stets die Gleichheit

$$a \oplus b = b \oplus a$$

gilt. Wie heisst diese Eigenschaft?

Aufgabe 2.31

Sei a eine beliebige binäre Folge (denken Sie sich z.B. $a = 1100101$).

(a) Was macht die Verschlüsslung $a \oplus a$ von a mit sich selbst?

(b) Mit 0 bezeichnen wir im Folgenden eine Folge aus lauter Nullen derselben Länge wie a. Was macht die Operation $a \oplus 0$?

Es lässt sich beweisen, dass die Operation $\oplus$ auch *assoziativ* ist. Die Klammerung der Terme spielt also keine Rolle. Für beliebige binäre Folgen a, b und c derselben Länge gilt also

$$(a \oplus b) \oplus c = a \oplus (b \oplus c).$$

Dies gilt auch für mehr als drei Operanden.

Aufgabe 2.32

Es sei t ein gegebener (binärer) Klartext. Alice wählt zufällig einen binären Schlüssel s_A derselben Länge wie t und berechnet

$$k_A := t \oplus s_A.$$

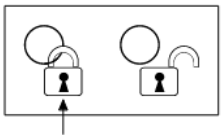
Was erhält Alice, wenn sie nun

$$k_A \oplus s_A$$

berechnet, also ihren Schlüssel erneut anwendet?

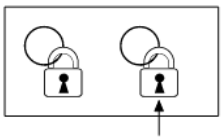
Kapitel 3

Schlüsseltausch-Verfahren



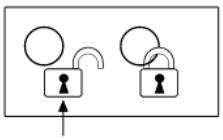
Absender schließt

	Klartext	101011
$\oplus$	Absender-Schlüssel	011011
	Erster Krypto-Text	110000



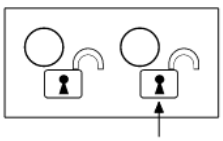
Empfänger schließt

	Erster Krypto-Text	110000
$\oplus$	Empfänger-Schlüssel	101010
	Zweiter Krypto-Text	011010



Absender öffnet

	Zweiter Krypto-Text	011010
$\oplus$	Absender-Schlüssel	011011
	Dritter Krypto-Text	000001



Empfänger öffnet

	Dritter Krypto-Text	000001
$\oplus$	Empfänger-Schlüssel	101010
	Klartext	101011

Abbildung 3.1: Schlüsseltausch

3.1 Drei-Wege-Schlüsseltausch

Vorgehen 3.1 (Kommunikationsprotokoll Three-Pass Protocol)**:**
Das Kommunikationsprotokoll Three-Pass Protocol (auch Schlüsseltausch mit drei Durchgängen genannt) ermöglicht es Alice, einen geheimen Schlüssel t an Bob zu senden, ohne dass sie zuvor einen gemeinsamen geheimen Schlüssel vereinbart haben. Dazu verwenden sowohl Alice als auch Bob jeweils einen eigenen zufälligen Schlüssel (s_A bzw. s_B).

Ausgangssituation: Alice besitzt einen zufälligen binären Schlüssel s_A der Länge n. Bob besitzt ebenfalls einen zufälligen binären Schlüssel derselben Länge n.

Ziel: Alice hat zuvor einen geheimen binären Schlüssel t der Länge n gewählt. Diesen Schlüssel (das ist hier der Klartext) möchte sie über einen unsicheren Kanal an Bob senden, ohne dass Unbefugte den Schlüssel erfahren.

1. **A**lice verschlüsselt den zu verschickenden Schlüssel t (Klartext) mit ihrem Schlüssel s_A

$$k_A := t \oplus s_A$$

   und sendet den entstandenen Kryptotext k_A an Bob.
2. **B**ob verschlüsselt die empfangene Nachricht k_A nun auch mit seinem Schlüssel s_B:

$$k_{AB} := k_A \oplus s_B$$

   und sendet den Kryptotext k_{AB} zurück an Alice.
3. Alice entschlüsselt den Kryptotext k_{AB} mit ihrem Schlüssel s_A:

$$k_B = k_{AB} \oplus s_A$$

   und sendet k_B an Bob.
4. Schliesslich entschlüsselt Bob den Kryptotext k_B mit seinem Schlüssel k_B:

$$t = k_B \oplus s_B$$

   und erhält dadurch den Klartext t, welchen ihn Alice wissen lassen möchte.

Warum funktioniert das? Der Kern der Geschichte liegt darin, dass eine zweite Anwendung eines Schlüssels die erste Anwendung desselben Schlüssels löscht (rückgängig macht) und zwar, auch wenn zwischen diesen zwei Anwendungen andere Schlüssel angewendet worden sind. Betrachten Sie die folgende Berechnung

$$\begin{aligned} &t \oplus s_A \oplus s_B \oplus s_A \oplus s_B = \\ &t \oplus (s_A \oplus s_A) \oplus (s_B \oplus s_B) = \\ &t \oplus 0 \oplus 0 = t. \end{aligned}$$

Aufgabe 3.1

Spielen Sie den Schlüsseltausch mit einer weiteren Person aus der Klasse durch.

Sicherheit des Drei-Wege-Schlüsseltauschs

Ist das Drei-Wege-Kommunikationsprotokoll sicher? Bietet es dieselbe Sicherheitsgarantie wie die Verwendung einer Truhe mit zwei Schlössern?

Wenn ein Kryptoanalyst nur einzelne Kryptotexte des Protokolls erhält und das Verfahren nicht kennt, wirkt die Kommunikation als eine Folge von Zufallsbits und in diesem Sinne ist unsere Implementierung dieses Verfahrens sicher. Wir müssen aber damit rechnen, dass die Gegnerin das Kommunikationsprotokoll kennt (oder erratet) und die zwei zufällig generierten Schlüssel das Einzige sind, was ihm unbekannt ist.

Wenn die Gegnerin (Eve) alle drei Kryptotexte (k_A, k_{AB}, k_B) gewinnen kann, kann er durch die folgenden Berechnungen den Klartext t herausfinden:

$$
\begin{aligned}
& k_A \oplus k_{AB} \oplus k_B = \\
& (t \oplus s_A) \oplus (k_A \oplus s_B) \oplus (k_{AB} \oplus s_A) = \\
& (t \oplus s_A) \oplus (k_A \oplus s_B) \oplus ((k_A \oplus s_B) \oplus s_A) = \\
& t \oplus (s_A \oplus s_A) \oplus (s_B \oplus s_B) \oplus (k_A \oplus k_A) = \\
& t \oplus 0 \oplus 0 \oplus 0 = \\
& t.
\end{aligned}
$$

3.2 Diffie-Hellman-Merkle-Schlüsseltausch

Vorgehen 3.2 (Protokoll Diffie-Hellman-Merkle (DHM))**:**
Das Kommunikationsprotokoll DHM ermöglicht es Alice und Bob, gemeinsam über einen öffentlichen Kanal einen geheimen Schlüssel zu vereinbaren, ohne dass sie zuvor einen gemeinsamen geheimen Schlüssel ausgemacht haben. Im Gegensatz zum Drei-Wege-Schlüsseltausch basiert das DHM-Protokoll auf der Schwierigkeit, das diskrete Logarithmusproblem zu lösen, ein mathematisches Problem, zu dem es (bisher) keinen effizienten Lösungsalgorithmus gibt.

Ausgangssituation: Alice und Bob haben sich zuvor öffentlich auf eine grosse Primzahl p und eine positive natürliche Zahl g geeinigt. Dabei ist g kleiner als p.

Ziel: Alice und Bob möchten gemeinsam mit einer öffentlichen Kommunikation einen Schlüssel s_{AB} vereinbaren. Diesen Schlüssel darf keine Drittperson in Erfahrung bringen.

1. Alice wählt zufällig eine positive ganze Zahl a mit $a < p$ und hält diese geheim. Dann berechnet sie mit dieser geheimen Zahl:
$$x := g^a \mod p$$
und sendet x an Bob.
2. Bob wählt zufällig eine positive ganze Zahl b mit $b < p$ und hält diese geheim. Dann berechnet er die Zahl
$$y := g^b \mod p$$
und sendet y an Alice.
3. Alice erhält y von Bob und berechnet mit ihrer geheimen Zahl a die Zahl
$$s_{AB} := y^a \mod p.$$
4. Bob berechnet mit dem erhaltenen x und seiner geheimen Zahl b die Zahl
$$s_{BA} := x^b \mod p.$$

Mithilfe der Rechengesetze der Modulo-Operation kann bewiesen werden, dass tatsächlich
$$s_{AB} = s_{BA}$$
gilt und somit Alice und Bob dieselbe Zahl berechnet haben. Diese Zahl s_{AB} ist der gemeinsame Schlüssel.

Aufgabe 3.2

Führen Sie das DHM-Protokoll mit einer weiteren Person aus der Klasse durch. Verwenden Sie
$$p := 13 \quad \text{und} \quad g := 2$$
(Sie können auch eigene Werte wählen) als öffentlich bekannte Schlüssel.

Aufgabe 3.3

Versuchen Sie das Kommunikationsprotokoll DHM zu knacken, indem Sie für die gegebenen Werte p, g, x und y jeweils die geheimen Zahlen a und b berechnen und daraus dann den vereinbarten Schlüssel s_{AB}.

(a) $g = 3, p = 5, x = 4, y = 2$
(b) $g = 2, p = 13, x = 6, y = 11$

Sicherheit des DHM-Protokolls

Das DHM-Protokoll ist sicher, weil es (bisher) keinen effizienten Algorithmus gibt, um das diskrete Logarithmusproblem zu lösen. Ein Kryptoanalyst (Eve) kann zwar die öffentlichen Werte p, g, x und y in Erfahrung bringen, aber um daraus den gemeinsamen Schlüssel s_{AB} zu berechnen, müsste sie entweder die geheime Zahl a von Alice oder die geheime Zahl b von Bob bestimmen. Dies entspricht dem Lösen des diskreten Logarithmusproblems, was (bisher) als schwierig gilt.

Allerdings ist das DHM-Protokoll nicht gegen einen Man-in-the-Middle-Angriff geschützt. Ein Angreifer (Eve) könnte sich zwischen Alice und Bob schalten und so tun, als ob sie Alice wäre, wenn sie mit Bob kommuniziert, und umgekehrt. Dadurch könnte Eve sowohl mit Alice als auch mit Bob jeweils einen eigenen gemeinsamen Schlüssel vereinbaren und so die Kommunikation zwischen den beiden belauschen. Diese Schwäche kann durch die Verwendung von digitalen Signaturen behoben werden, welche die Authentizität der Kommunikationspartner sicherstellen. Ein Beispiel eines solchen Signaturverfahrens ist das Rivest–Shamir–Adleman (RSA)-Kryptosystem (s. Kapitel Abschnitt 4.1).

Kapitel 4

Asymmetrische Kryptosysteme

In den bisher angeschauten Verschlüsslungsverfahren haben wir festgestellt, dass derselbe Schlüssel zur Ver- und Entschlüsselung verwendet wird. Daher spricht man bei diesen Verfahren von *symmetrischen* Verschlüsselungsmethoden. Zudem haben wir gesehen, dass diese entweder unsicher (Caesar, Vigenère) sind, wenn der Schlüssel einfach geknackt werden kann, oder un-praktikabel (OTP), wenn der Schlüssel zuerst übertragen werden muss. Diffie & Hellman kamen daher 1975 auf die Idee, *asymmetrische* Verschlüsselungsverfahren zu erschaffen, welche nach einem anderen Prinzip funktionieren. Im Gegensatz zu symmetrischen Verfahren kann man bei asymmetrischen Verfahren *nicht* von der verschlüsselten Nachricht auf den Schlüssel schliessen, da unterschiedliche Schlüssel zum Ver- und Entschlüsseln verwendet werden.

Die Grundidee hinter asymmetrischen Verschlüsselungsmethoden ist folgende:

Alice (👩‍💻) und Bob (👨‍💻) möchten auf verschlüsselte Weise Nachrichten austauschen, die den Anforderungen an sichere Kryptosysteme genügen (siehe Abschnitt 1.1). Bei der asymmetrischen Verschlüsselung generieren sowohl Alice wie Bob jeweils ein Schlüsselpaar, einen sogenannten **öffentlichen Schlüssel** (🔑), der von allen Personen gesehen und verwendet werden kann, sowie einen **privaten Schlüssel** (🔒), der nur im Besitz von Alice bzw. Bob ist. Wenn Alice eine Nachricht an Bob senden will, verwendet sie Bobs *öffentlichen* Schlüssel, um die Nachricht zu verschlüsseln. Die Nachricht kann jedoch mit dem öffentlichen Schlüssel nicht entschlüsselt werden, es handelt sich hier sozusagen um eine “Einweg”-Funktion. Stattdessen muss Bob seinen *privaten* Schlüssel zur Entschlüsselung verwenden, also den Schlüssel, auf den *nur* Bob Zugriff hat (siehe Abbildung 4.1). Dies funktioniert in den meisten Fällen auch in die andere Richtung: eine Nachricht, die mit Bobs privatem Schlüssel verschlüsselt worden ist, kann nur mit Bobs öffentlichem Schlüssel entschlüsselt werden. Die Schlüssel sind mathematisch so konstruiert, dass es beinahe unmöglich ist, vom öffentlichen Schlüssel auf den privaten Schlüssel zu schliessen.

Durch den Aufbau asymmetrischer Verschlüsselungsverfahren entfällt die Problematik der Übermittlung des Schlüssels: jede Person generiert ihr eigenes Schlüsselpaar und stellt einen öffentlichen Schlüssel zur Verfügung. Ein Nachteil hierbei ist, dass die Verschlüsselung häufig mathematisch und bezüglich Rechenleistung anspruchsvoller ist, was insbesondere problematisch sein kann bei längeren Nachrichten oder wenn die Antwortzeit minimal sein soll.

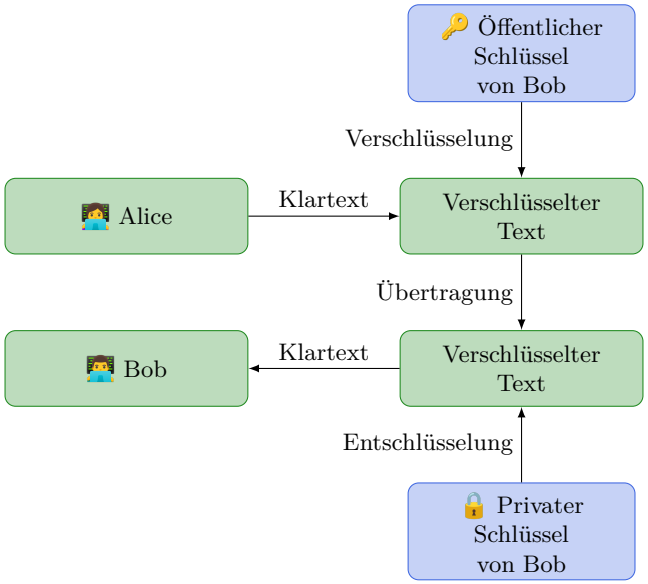


Abbildung 4.1: Prinzip der Public-Key-Verschlüsselung

4.1 RSA-Verfahren

Das Rivest–Shamir–Adleman (RSA)-Verfahren ist ein asymmetrisches Kryptosystem, das sowohl für die Verschlüsselung als auch für digitale Signaturen verwendet werden kann. Es basiert auf der Schwierigkeit, grosse Zahlen in ihre Primfaktoren zu zerlegen. Im Folgenden werden die Grundlagen des Verfahrens und ein einfaches Beispiel vorgestellt.

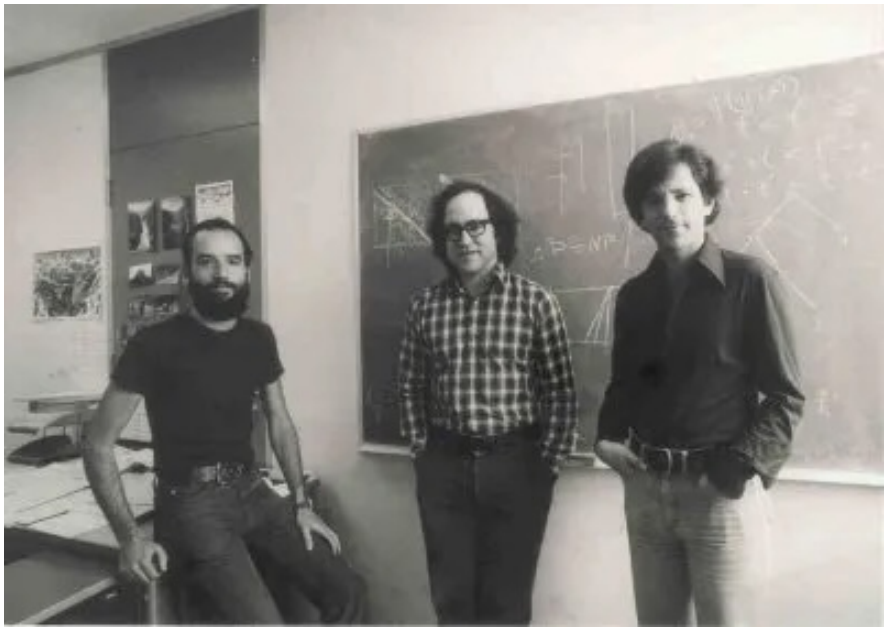


Abbildung 4.2: Ron Rivest, Adi Shamir und Leonard Adleman, die Erfinder des RSA-Verfahrens

Vorgehen 4.1 (RSA-Verschlüsselungsverfahren)**:**
Das RSA-Verfahren besteht aus folgenden Schritten:

1. **Schlüsselerzeugung:**

- Wählen Sie zwei (möglichst grosse) Primzahlen p und q.
- Berechnen Sie das RSA-Modul $n = p \cdot q$.
- Berechnen Sie die Eulersche Funktion $\varphi(n) = (p-1)(q-1)$.
- Wählen Sie den *Verschlüsslungsexponenten* e, so dass dieser teilerfremd zu $\varphi(n)$ und kleiner als $\varphi(n)$ ist. Dies bedeutet, dass der Grösster Gemeinsamer Teiler (GGT) von e und $\varphi(n)$ 1 ist ($\text{ggT}(e, \varphi(n)) = 1$).
- Berechnen Sie den *Entschlüsslungexponenten* d, so dass $(e \cdot d) \mod \varphi(n) = 1$ (privater Schlüssel). Dies bedeutet, dass, wenn man d mit e multipliziert und dieses Produkt Modulo $\varphi(n)$ rechnet, man die Zahl 1 erhält.
- Die Zahlen p, q und $\varphi(n)$ werden nun nicht mehr benötigt und können gelöscht werden.

2. **Verschlüsselung:**
   - Der Absender verschlüsselt eine **Klartext-Nachricht m** (als Zahl) mit dem **öffentlichen Schlüssel** $(\mathbf{e}, \mathbf{n})$:
   $$c = \mathbf{m}^e \mod n$$
   - Das Ergebnis c ist der Kryptotext.
3. **Entschlüsselung:**
   - Der Empfänger entschlüsselt die **verschlüsselte Nachricht c** mit dem **privaten Schlüssel** $(\mathbf{d}, \mathbf{n})$:
   $$\mathbf{m} = c^d \mod n$$
   - Das Ergebnis **m** ist die ursprüngliche Nachricht.

Beispiel 4.1:
Um RSA besser zu verstehen, rechnen wir ein einfaches Beispiel mit kleinen Zahlen durch:

1. Wir wählen zwei (für diese Übung kleine) Primzahlen aus, $p = 3$ und $q = 11$. Daraus folgt:
$$\begin{aligned} n &= p \cdot q \\ &= 3 \cdot 11 \\ &= 33 \end{aligned}$$
$$\begin{aligned} \varphi(n) &= (p-1)(q-1) \\ &= 2 \cdot 10 \\ &= 20 \end{aligned}$$
2. Wir wählen $e = 3$, da $\text{ggT}(3, 20) = 1$.
3. Wir berechnen d, so dass $(e \cdot d) \mod \varphi(n) = 1$ ergibt:
$$(d \cdot 3) \mod 20 = 1 \quad \rightarrow \quad d = 7.$$
4. Der öffentliche Schlüssel ist $(e, n) = (3, 33)$, der private Schlüssel $(d, n) = (7, 33)$.

Verschlüsselung: Die Nachricht sei das Wort "Code", welches wir darstellen durch die Position der Buchstaben im Alphabet $m = 3, 15, 4, 5$. Berechne für jeden Buchstaben (hier

nur für “C”, also 3, gezeigt):

$$\begin{aligned} c &= m^e \mod n \\ &= 3^3 \mod 33 \\ &= 27 \mod 33 \\ &= 27. \end{aligned}$$

Der erste Buchstabe des Kryptotext wird also verschlüsselt als $c = 27$.

Entschlüsselung:

$$\begin{aligned} m &= c^d \mod n \\ &= 27^7 \mod 33 \\ &= 3 \end{aligned}$$

Daraus ergibt sich $m = 3$, also die ursprüngliche Nachricht.

Aufgabe 4.1

Alice möchte eine Nachricht an Bob mit RSA verschlüsseln. Bob wählt die Hilfs-Primzahlen $p = 5$ und $q = 7$. Generieren Sie den öffentlichen Schlüssel (e, n) und den privaten Schlüssel (d, n) von Bob, indem Sie folgende Schritte ausführen:

1. Berechnen Sie n und $\varphi(n)$.
2. Sie wählen e und d aus.

Verschlüsseln Sie nun die Nachricht $m = 9$ mit dem öffentlichen Schlüssel (e, n).

Entschlüsseln Sie danach die verschlüsselte Nachricht c wieder mit dem privaten Schlüssel (d, n).

Aufgabe 4.2

Ist die Wahl von p und q im obigen Beispiel sinnvoll? Begründen Sie Ihre Antwort.

Aufgabe (Challenge) 4.3

Erstellen Sie nun zu zweit jeweils ein Schlüsselpaar mit dem RSA-Verfahren. Verwenden Sie dazu Primzahlen p und q mit jeweils mindestens 2 Ziffern. Tauschen Sie danach Ihre öffentlichen Schlüssel aus und verschlüsseln Sie eine Nachricht (eine Zahl) an Ihren Partner. Entschlüsseln Sie danach die Nachricht wieder.

Eine Liste der ersten 1000 Primzahlen finden Sie hier: https://en.wikipedia.org/wiki/List_of_prime_numbers.

Aufgabe (Challenge) 4.4

Verwenden Sie Ihre RSA-Schlüssel aus 4.3, um eine echte Nachricht als Zahl auszutauschen (ein Wort). Um ein Wort in Zahlen umzuwandeln (Buchstabe für Buchstabe) können Sie folgendes Python-Skript verwenden:

```
def text_to_numbers(text):
    numbers = []
    for char in text.upper():
        if char.isalpha():  # Nur Buchstaben berücksichtigen
            numbers.append(ord(char) - ord('A') + 1)  # A=1, B=2, ..., Z=26
    return numbers

print(text_to_numbers("ABC"))  # Ausgabe: [1, 2, 3]
```

Aufgabe (Challenge) 4.5

Wählen Sie zwei Primzahlen p und q mit jeweils mindestens 3 Ziffern. Generieren Sie den öffentlichen und privaten Schlüssel. Verschlüsseln Sie eine Nachricht Ihrer Wahl und entschlüsseln Sie diese wieder. Verwenden Sie Python, um die Berechnungen durchzuführen.

Die Zahl d kann in Python folgendermassen berechnet werden:

Eine Liste der ersten 1000 Primzahlen finden Sie hier: https://en.wikipedia.org/wiki/List_of_prime_numbers

```
d = pow(e, -1, phi_n)
```

4.1.1 Digitale Signaturen mit RSA

Neben der Verschlüsselung von Nachrichten kann das RSA-Verfahren auch zur Erstellung digitaler Signaturen verwendet werden. Digitale Signaturen dienen dazu, die Authentizität und Integrität einer Nachricht zu gewährleisten. Hierbei wird die Nachricht mit dem privaten Schlüssel des Absenders signiert, sodass der Empfänger die Signatur mit dem öffentlichen Schlüssel des Absenders überprüfen kann.

Vorgehen 4.2 (Digitale Signaturen mit RSA)**:**
Die Erstellung und Überprüfung digitaler Signaturen mit dem RSA-Verfahren erfolgt in folgenden Schritten:

1. **Signaturerstellung:**
   - Der Absender erstellt eine **Nachricht m**.
   - Er berechnet den **Hashwert** $H(m)$ der Nachricht m mithilfe einer kryptographischen Hashfunktion (z.B. SHA-256).
   - Der Absender signiert den Hashwert mit seinem **privaten Schlüssel** $(\mathbf{d}, \mathbf{n})$:

$$s = H(m)^d \mod n$$

   - Das Ergebnis s ist die digitale Signatur.
2. **Signaturüberprüfung:**
   - Der Empfänger erhält die Nachricht m und die Signatur s.

- Er berechnet den Hashwert $H(m)$ der empfangenen Nachricht m.
- Der Empfänger überprüft die Signatur mit dem **öffentlichen Schlüssel** $(\mathbf{e}, \mathbf{n})$ des Absenders:
$$H'(m) = s^e \mod n$$
- Wenn $H'(m) = H(m)$, ist die Signatur gültig, andernfalls ist sie ungültig.

Beispiel 4.2:
Angenommen, Alice möchte eine Nachricht $m = 42$ signieren. Sie verwendet ihren privaten Schlüssel $(d, n) = (9677, 12317)$ und den öffentlichen Schlüssel $(e, n) = (5, 12317)$.

Signaturerstellung:

$$\begin{aligned} s &= H(m)^d \mod n \\ &= 42^{9677} \mod 12317 \\ &= 161 \end{aligned}$$

Die digitale Signatur ist also $s = 161$.

Signaturüberprüfung:

$$\begin{aligned} H'(m) &= s^e \mod n \\ &= 161^5 \mod 12317 \\ &= 42 \end{aligned}$$

Da $H'(m) = H(m)$, ist die Signatur gültig.

Aufgabe 4.6

Alice möchte die Nachricht $m = 15$ signieren. Ihr privater Schlüssel ist $(d, n) = (7, 33)$ und ihr öffentlicher Schlüssel ist $(e, n) = (3, 33)$.

Erstellen Sie die digitale Signatur s für die Nachricht m.

Überprüfen Sie danach die Signatur mit dem öffentlichen Schlüssel.

Anhang A

Python-Übungen zu Kryptologie

A.1 Allgemeine Zeichenketten-Aufgaben

Zeichenketten können verkettet, also aneinandergehängt werden mit dem Befehl `"Text1" + "Text2" + "Text3"` usw. Falls Sie eine Zeichenkette mehrfach drucken wollen, können Sie diesen mit einer Zahl multiplizieren, die dann die Anzahl Wiederholungen der Zeichenkette bestimmt. So ist der Ausdruck `"a" * 3` beispielsweise gleichbedeutend mit einer Zeichenkette `"aaa"`.

Im Nachfolgenden schauen wir uns einige Übungen an, mit denen wir die einzelnen Zeichen aus Zeichenketten herauslesen können.

Aufgabe A.1

Führen Sie das nachfolgende Programm aus und erklären Sie, was das Programm tut.

```
text = "EASY"
for buchstabe in text:
  print(buchstabe)
```

Aufgabe A.2

Führen Sie das nachfolgende Programm aus und erklären Sie, was das Programm tut.

```
Klartext = "Schweiz"
print(Klartext[0])
print(Klartext[1])
print(Klartext[2])
print(Klartext[3])
print(Klartext[4])
print(Klartext[5])
print(Klartext[6])
```

Aufgabe A.3

Führen Sie das nachfolgende Programm aus und erklären Sie, was das Programm tut.

```
Klartext = "Schweiz"
for i in range(len(Klartext)):
  print(Klartext[i])
```

Der Befehl `for i in range(len(Klartext))` erstellt eine Variable `i` innerhalb der `for`-Schleife, welche jedes Zeichen von 0 bis zur Länge von `Klartext` minus 1 geht. Weshalb minus 1? Das erste Zeichen von `Klartext` wird in Python mit `Klartext[0]` ausgelesen, das letzte mit `Klartext[len(Klartext)-1]`, da wir bei 0 zu zählen beginnen und nicht bei 1. Mit dem Ausdruck `Klartext[i]` wird also das `i`-te Zeichen der Zeichenkette `Klartext` ausgelesen, wobei `i` von 0 bis zu (Länge des Klartexts minus 1) geht.

Der Befehl `for i in range(len(Klartext))` kann auch geschrieben werden also Befehl `for i in range(0, len(Klartext), 1)`, wobei dies meint:

- `i` beginnt bei 0
- `i` geht bis zu `len(Klartext)-1`
- `i` vergrössert sich in 1er-Schritten

Dieser Befehl könnte auch verwendet werden, um bei einer beliebigen Zahl zu starten (nicht notwendigerweise 0), und um `i` in Schritten grösser als 1 zu vergrössern. Die allgemeine Syntax des Befehls lässt sich also wie folgt zusammenfassen: `for i in range(start, ende, schrittgroesse)`.

Aufgabe (Challenge) A.4

Eine Person verrät uns lediglich die Vorwahl ihrer 10-stelligen Mobiltelefonnummer. Zudem verrät Sie Ihnen auch, dass ihre Telefonnummer gerade ist (also auf 2, 4, 6, 8 oder 0 endet). Damit gibt es nur noch 5 Millionen Kombinationen, die es (im schlimmsten Fall) auszuprobieren gilt. Schreiben Sie ein Python-Programm, welches die 200 kleinsten, geraden Nummern auflistet. Die erste Nummer sollte 0790000000 sein, die letzte 0790000198. Das Programm soll die Nummern als Zeichenkette (eine Nummer pro Zeile) ausgeben.

Tipps:

- Mit `str(num)` wird aus der Zahl `num` eine Zeichenkette. Beispielsweise gibt uns `str(15)` die Zeichenkette `"15"`.
- Erinnern Sie sich, was die Operation `+` in dem Ausdruck `"In" + "form" + "atik"` macht?

A.2 Verschlüsselung von Texten in Python

Während des Zweiten Weltkriegs arbeitete der brillante Mathematiker Alan Turing in Bletchley Park im Vereinten Königreich und war mit der entscheidenden Aufgabe betraut, den Enigma-Code zu knacken, den die Deutschen zur Verschlüsselung ihrer Nachrichten verwendeten. Um diesen komplexen Code zu entschlüsseln, musste Turing sein tiefes Verständnis von Sprache und Mustern nutzen. In diesem Kapitel folgen wir in Turings Fussstapfen und entziffern einige Geheimnachrichten!

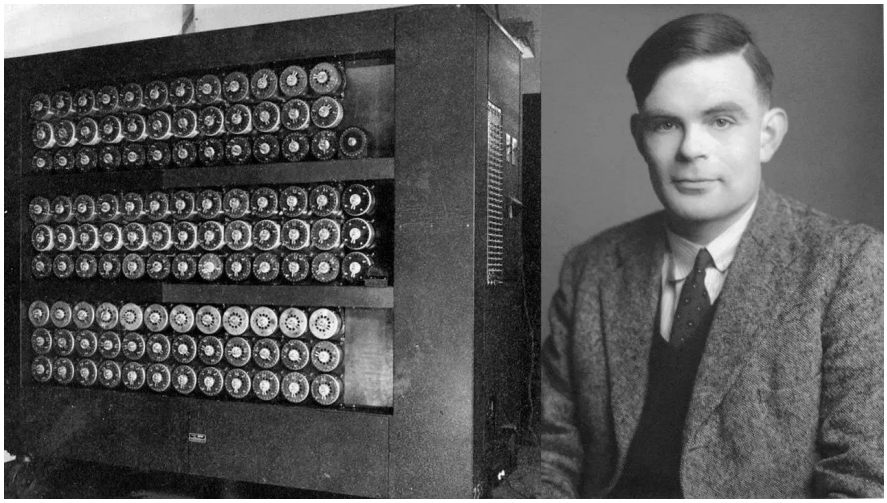


Abbildung A.1: Turing mit seiner berühmten Turing-Maschine

Im Nachfolgenden schauen wir uns einige Python-Befehle an, die dazu dienen, mit Zeichenketten (= engl. *strings*) zu arbeiten. Dies werden wir insbesondere benötigen, um eigene Python-Programme zu schreiben, mit denen wir Texte verschlüsseln und entschlüsseln können. Genauer gesagt bezeichnen wir Texte in Python als Zeichenketten. Zeichenketten werden in Python innerhalb von einfachen oder doppelten Anführungszeichen geschrieben, also entweder `'...'` oder `"..."`. Innerhalb der Anführungszeichen können beliebige Zeichen stehen, beispielsweise Buchstaben, Leerzeichen, Spezialzeichen oder Zahlen.

Tabelle A.1 und Tabelle A.2 enthalten eine Übersicht einiger nützlicher Befehle, die Sie in diesem Kapitel verwenden werden.

Python	Beschreibung	Beispiel
`len(s)`	Gibt die Länge des Textes `s` zurück.	`n = len("hallo") # 5`

`s[index]`	Greift auf das Zeichen an der Index-Position `index` zu. Bei negativen Indizes zählt man von rechts.	`s = "hallo"`
		`s = "hallo"`
		`c0 = s[0]  # "h"`
		`c1 = s[1]  # "a"`
		`c2 = s[-1] # "o"`

`s[start:end]`	Extrahiert den Teiltext vom Index `start` bis und ohne `end`	`s1 = "hallo"` `s2 = s1[1:4] # "all"`

`ord(c)`	Gibt die Position des Zeichens `c` in der Unicode-Tabelle zurück	`n1 = ord('A') # 65`
		`n2 = ord('B') # 66`
		`n3 = ord('Z') # 90`

`chr(n)`	Gibt die das Zeichen an der `n`-ten Position er Unicode-Tabelle zurück	`c1 = chr(65) # "A"`
		`c2 = chr(66) # "B"`
		`c3 = chr(67) # "C"`

`s1.count(s2)`	Zählt, wie oft der Text `s2` im Text `s1` vorkommt	`s1 = "Das Haus ist gross und das Dach ist grün."`
		`s2 = "ist"`
		`n = s1.count(s2) # 2`

`s1.find(s2)`	Gibt den niedrigsten Index des Teiltexts `s2` zurück, falls dieser vorhanden ist, ansonsten `-1`	`s = "Das Haus ist gross und das Dach ist grün."` `n = s.find("ist") # 9`
`s1.replace(old, new)`	Ersetzt alle Vorkommen des Teil-Texts `old` durch `new`.	`s1 = "hallo"` `s2 = s1.replace("a", "e") # "hello"`

Tabelle A.1: Übersicht von Zeichenketten-Befehlen

Aufgabe A.5

Schreiben Sie ein Programm, um jeden Buchstaben „e / E“ des folgenden Texts durch den Buchstaben durch „a / A“ zu ersetzen:

```
my_text = "Eines Tages entschied der Elefant, einen edlen Teppich zu weben."
```

Aufgabe A.6

Schreiben Sie ein Programm, um zu zählen, wie häufig der Buchstabe „e / E“ im gesamten folgenden Text vorkommt:

```
my_text = "Eines Tages entschied der Elefant, einen edlen Teppich zu weben."
```

Aufgabe (Challenge) A.7

Was könnte der Nutzen davon sein, dass es zwei Möglichkeiten gibt und nicht nur eine, Zeichenketten zu schreiben (`'...'` oder `"..."`)?

Mit der **Länge** von Zeichenketten ist die Anzahl der Zeichen zwischen den Anführungszeichen gemeint: Die Zeichenkette `"Hello World"` hat beispielsweise eine Länge von 11 Zeichen. Die Länge einer Zeichenkette kann mit dem Ausdruck `len(Klartext)` bestimmt werden.

Aufgabe A.8

Bestimmen Sie die Länge Ihres vollen Namens mithilfe des Befehls `len()` in Python.

```
n = "Vorname Nachname"
print(len(n)) # 16
```

Die Lösung für Aufgabe 2.1 könnte in Python wie folgt implementiert werden:

```
def zweiertausch(klartext):
    geheimtext = ""  # leerer String
    b = 0
    while b < (len(klartext) - 1):
        geheimtext += klartext[b + 1] + klartext[b]
        b += 2

    if len(klartext) % 2 != 0:
        # Klartexte ungerader Länge
        geheimtext += klartext[len(klartext) - 1]


print(geheimtext)

# Verwendung:
# zweiertausch("KANTONSSCHULE")
```

Aufgabe A.9

Schreiben Sie eine Python-Funktion `def dreiertausch(klartext)`, welche den „Dreiertausch“ aus der Einführungsaufgabe 2 implementiert.

Aufgabe A.10

Schreiben Sie eine Python-Funktion `def umkehren(klartext)`, welche alle Zeichen eines Klartexts in umgekehrter Reihenfolge ausgibt.

Python	Beschreibung	Beispiel

`s1.split(sep)`	Zerlegt den Text an jedem `sep` und gibt eine Liste der Teiltexte zurück.	`s = "a,b,c"`
		`li = s.split(",")`
		`# ["a", "b", "c"]`

`s.join(li)`	Verbindet die Elemente von `li` zu einem Text, getrennt durch den Text `s`.	`txt = ":".join(["a", "b", "c"])` `# "a:b:c"`
`str(n)`	Zahl `n` in einen Text (engl. *string*) umwandeln	`n = 10` `s = str(n) # "10"`
`int(s)`	Text `s` in eine ganze Zahl (engl. *integer*) umwandeln	`s = "10"` `n = int(s) # 10`

Tabelle A.2: Übersicht von Befehlen, um Zeichenketten zu verbinden, bzw. trennen

Aufgabe A.11

Der Schlüssel für einen Geheimtext ist in Blöcken organisiert:

```
geheime_zahl_als_text = "2_10_38"
```

Trennen Sie die Blöcke von `geheime_zahl_als_text` in eine Liste auf und addieren Sie die Zahlen der Liste zusammen.

Als Resultat sollten Sie die Zahl 50 erhalten.

Aufgabe A.12

Verbinden Sie die Wörter in der Liste `liste = ["Der", "Code", "wurde", "geknackt"]` mit Leerzeichen zu einem vollständigen Satz.

A.3 Caesar-Verschlüsselung in Python

Aufgabe A.13

Entwickeln Sie eine Funktion, die zwei Parameter entgegennimmt: den Klartext (nur Grossbuchstaben!) und den Schlüssel (0-25). Als erstes soll jeder Buchstaben des Klartexts auf einer neuen Zeile ausgegeben werden.

Beispiel: Falls der Text `HALLO` eingegeben wird, soll folgendes ausgegeben werden:

```
H
A
L
L
O
```

Aufgabe A.14

Passen Sie das Programm aus Aufgabe 1.13 so an, dass statt den Buchstaben die Position in der Unicode-Tabelle ausgegeben wird. Verwenden Sie dafür die Funktion `ord()`.

Beispiel: Falls der Text `HALLO` eingegeben wird, soll folgendes ausgegeben werden:

```
72
65
76
76
79
```

Aufgabe A.15

Passen Sie das Programm aus Aufgabe 1.14 so an, dass Sie zu den Unicode-Positionen auch noch die Verschiebung hinzurechnen.

Beispiel: Falls der Text `HALLO` eingegeben wird und die Verschiebung 2, soll folgendes ausgegeben werden:

```
74
67
78
78
81
```

Aufgabe A.16

Passen Sie das Programm aus Aufgabe 1.15 so an, dass Sie statt den verschobenen Unicode-Positionen die Buchstaben ausgeben. Verwenden Sie dafür die Funktion `chr()`.

Beispiel: Falls der Text `HALLO` eingegeben wird und die Verschiebung 2, soll folgendes ausgegeben werden:

```
J
C
N
N
Q
```

Aufgabe A.17

Wenn man den Klartext `ZORRO` mit dem Schlüssel 14 verschlüsselt, erhält man mit dem Programm aus Aufgabe Aufgabe 1.16 den Geheimtext `h]``]`. Eigentlich sollte man aber den Geheimtext `NCFFC` erhalten. Lösen Sie das Problem!

Tipp: Der grösste gültige Unicode-Wert ist 90 für den Buchstaben „Z“. Wenn man einen Wert bekommt, der 91 oder grösser ist, muss man ihn verkleinern!

Aufgabe A.18

Passen Sie das Programm aus Aufgabe 1.17 so an, dass die Geheimtextbuchstaben nicht untereinander, sondern auf derselben Zeile in die Ausgabe geschrieben werden.

Tipp: Mit dem Operator `+` lassen sich in Python Texte miteinander verbinden.

```
text = ""
text += "Hello"
text += " Bob"
print(text) # Hello Bob
```

Die verschlüsselte Zeichenkette soll per `return`-Befehl zurückgegeben werden.

Aufgabe A.19

Entwickeln Sie ein Programm, das über einen Parameter den Geheimtext und den Schlüssel entgegennimmt und dann mit Hilfe der Caesar-Chiffre den Klartext wiederherstellt. Testen Sie es anschliessend, indem Sie eine verschlüsselte Nachricht von Ihrer Sitznachbarin oder Ihrem Sitznachbar entschlüsseln!

A.4 Vigenère-Verschlüsselung in Python

Aufgabe A.20

Entwickeln Sie eine Funktion `def vigenere(text, key)`, die über die zwei Parameter `text` und `key` einen Klartext sowie einen Schlüssel entgegennimmt. Das Programm soll den Klartext mit der Vigenère-Chiffre und mit dem Schlüssel verschlüsseln.

Aufgabe A.21

Entwickeln Sie eine Funktion `def vigenere_ent(text, key)`, die über die zwei Parameter `text` und `key` einen Geheimtext sowie einen Schlüssel entgegennimmt. Das Programm soll den Geheimtext mit der Vigenère-Chiffre und mit dem Schlüssel entschlüsseln.

Aufgabe (Challenge) A.22

Sie wollen all ihre Passwörter auf einer lokalen Textdatei abspeichern. Dies ist jedoch nicht sicher, denn falls jemand ihren Computer knackt, kann die Person alle Passwörter einsehen. Sie kommen auf folgende Idee: Statt die Passwörter aufzuschreiben, speichern Sie eine mit Vigenère verschlüsselte Variante davon. So müssen Sie sich lediglich den Vigenère-Schlüssel im Kopf merken, nicht aber ihre Passwörter. Verwenden Sie Ihr Programm aus Aufgabe 1.20, um ein Programm zu schreiben, das sie zuerst mit `input` nach einem Schlüssel und einem verschlüsselten Passwort fragt. Das Programm gibt Ihnen danach ihr Passwort im Klartext zurück.

Aufgabe A.23 Häufigkeitsanalyse und Caesar

Eine Nachricht wurde abgefangen: `msg = "AZDIYDHRZNOZI"`.

- Die Nachricht wurde mit der Caesar-Chiffre verschlüsselt. Finden Sie zunächst einmal den häufigsten Buchstaben in der verschlüsselten Nachricht. **Tipp**: „A“ und „Z“ haben die Positionen 65 und 90 im Unicode. Um in einer Schleife alle Zahlen von `x` by `y` durchzugehen, können Sie Folgendes schreiben: `for num in range(x, y+1):`.
- Bestimmen Sie den verwendeten Schlüssel, indem Sie die Funktion `ord()` zweimal verwenden.
- Ermitteln Sie den Klartext mithilfe Ihres Programms aus Aufgabe 1.18. **Tipp**: Das häufigste Zeichen in einem deutschen Text ist in der Regel „E“.
- Ersetzen Sie am Schluss im Klartext das Wort „WESTEN“ durch „OSTEN“.

Aufgabe A.24

Gegeben sei die durchschnittliche Buchstabenhäufigkeit für alle Buchstaben des deutschen Alphabets. Diese lässt sich berechnen, indem man die Häufigkeiten für sehr lange deutsche Texte aufsummiert.

```
german_letter_frequencies = [6.51, 1.89, 3.06, 5.08, 17.40, 1.66, 3.01,
    4.76, 7.55, 0.27, 1.21, 3.44, 2.53, 9.78, 2.51, 0.79, 0.02, 7.00, 7.27,
    6.15, 4.35, 0.67, 1.89, 0.03, 0.04, 1.13]
```

Berechnen Sie mithilfe folgenden Python-Programms die Friedman'sche Charakteristik für diese Häufigkeiten.

```
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick
import numpy as np
from helpers import count_letters


def calculate_fc(text):
    # Determine the Friedman Characteristic for a given text
    summe = 0
    freq = count_letters(text)
    for letter_freq in freq:
        summe += (letter_freq - (1 / 26)) ** 2
    return summe


def friedman_slice(text, keylength):
    # Based on a text encrypted with Friedman and a given keylength,
    determine the average Friedman characteristic for all subgroups of the
    text.
    i = 0
    fc_avg = 0
    for i in range(keylength):
        text_slice = text[i::keylength]
        fc = calculate_fc(text_slice)
        fc_avg += fc
        i += 1

    fc_avg /= keylength

    return fc_avg


def get_friedman_vals(text, maxkeylen):
    """
    For a Text text, get the average Friedman Characteristics for key
    lengths up to maxkeylen
    """
    n = range(1, maxkeylen)
```

```
    fc = []
    for i in n:
        fc.append(friedman_slice(text, i))
    return fc


def draw_friedman(i, fc, turtle=False):
    """
    Draw the friedman Characteristics for various possible key lengths
    """
    if turtle:
        xshift = 150
        setPos(-xshift + 50, 150)
        label("friedman'sche Charakteristik:")
        setPos(-xshift, 0)
        pd()
        fd(200)
        bk(200)
        rt(90)
        fd(400)
        bk(400)

        for i in n:
            setPos(i * 40 - xshift, fc[i - 1] * 1000)
            dot(10)
            setPos(i * 40 - xshift, fc[i - 1] * 1000 + 40)
            label(i)
    else:
        fig, ax = plt.subplots()
        ax.plot(range(1, i), fc, "o")
        plt.xticks(np.arange(1, i, 1.0))
        ax.yaxis.set_major_formatter(mtick.PercentFormatter(decimals=0, xmax
    =1))
        return fig, ax


if __name__ == "__main__":
    print(calculate_fc("PAPPERLAPAPP"))
    print(calculate_fc("BACKSTEIN"))
```

Anhang B

Lernziele Kryptologie

- ☐ Ich weiss wie die folgenden Kryptosysteme funktionieren (Verschlüsslung und Entschlüsslung bei gegebenem Schlüssel):
  - ☐ Skytale
  - ☐ Caesar
  - ☐ Vigenère
  - ☐ One-Time-Pad
  - ☐ RSA
- ☐ Ich kenne die Grundbegriffe: Klartext, Kryptotext, Verschlüsselung (Chiffrierung), Entschlüsselung (Dechiffrierung), Schlüssel, Kryptografie / Kryptologie, Kryptosystem.
- ☐ Ich kenne die grundlegenden Sicherheitsziele: Vertraulichkeit, Integrität, Authentizität und Verbindlichkeit (Nichtabstreitbarkeit) und kann sie kurz erläutern.
- ☐ Ich kann das Kerckhoff'sche Prinzip erklären.
- ☐ Ich kann einen Caesar-Kryptotext durch Ausprobieren aller 25 Verschiebungen knacken (Brute-Force) und erkenne den richtigen Klartext.
- ☐ Ich kann einen Kryptotext, der durch die Vigenère-Verschlüsslung entstanden ist, mit Hilfe des Tools auf folgender Webseite entschlüsseln: Link zum Cryptbreaker
- ☐ Ich kann die Idee der Häufigkeitsanalyse erklären.
- ☐ Ich kann eine gruppenweise Häufigkeitsanalyse durchführen, um Vigenère bei bekannter Schlüssellänge zu knacken.
- ☐ Ich kann erläutern, weshalb eine kurze Schlüssellänge Vigenère schwächt (Wiederholungsmuster, Aufteilung in Gruppen) und warum lange bzw. OTP-lange Schlüssel nicht angreifbar durch Häufigkeitsanalyse sind.
- ☐ Ich kann den Unterschied zwischen monoalphabetischer und polyalphabetischer Substitution erklären.
- ☐ Ich kann einen monoalphabetisch substituierten Text mittels Häufigkeitsanalyse und schrittweisem Erraten entschlüsseln.
- ☐ Ich kann die Friedman'sche Charakteristik für einen kurzen Text von Hand berechnen
- ☐ Ich kann mit der Friedman'schen Charakteristik die Länge eines Vigenère-Schlüssels bestimmen.
- ☐ Ich kann erklären, was mit der Friedman'sche Charakteristik berechnet wird.
- ☐ Ich kann beim One-Time-Pad begründen, warum es (bei perfekter Zufälligkeit, einmaliger Verwendung und gleicher Länge wie der Klartext) perfekte Sicherheit bietet (Schlüsselraum = Nachrichtenraum, Gleichverteilung der möglichen Klartexte).
- ☐ Ich kann die Anzahl möglicher OTP-Schlüssel einer gegebenen Länge berechnen (z.B. 26^n für Buchstaben, 2^n für Bits).
- ☐ Ich kann erklären, warum die Wiederverwendung eines OTP-Schlüssels zu Informationsleckage

führt und ein Beispiel analysieren.

- ☐ Ich kann die binäre Version des OTP (Bitweise XOR) anwenden und Kryptotexte berechnen.
- ☐ Ich kenne die Eigenschaften der XOR-Operation: Kommutativität, Assoziativität, neutrales Element 0, Involution (selbstinverse: $a \oplus a = 0$, $a \oplus 0 = a$) und deren Bedeutung für Ver- und Entschlüsselung.
- ☐ Ich kann erläutern, warum zweimalige Anwendung desselben Schlüssels (mit XOR) den Klartext zurückgibt.
- ☐ Ich kann das Three-Pass Protocol (Schlüsseltausch mit drei Durchgängen) Schritt für Schritt durchführen und die beteiligten Nachrichten (k_A, k_{AB}, k_B) bestimmen.
- ☐ Ich kann erklären, warum das Three-Pass Protocol unsicher ist, wenn ein Angreifer alle drei Kryptotexte mitschneidet (Rekonstruktion von t durch XOR aller Nachrichten).
- ☐ Ich kann das Diffie-Hellman-Merkle-Schlüsseltauschverfahren mit kleinen Zahlen durchführen und den gemeinsamen Schlüssel berechnen.
- ☐ Ich kann das zugrunde liegende Sicherheitsprinzip von Diffie-Hellman (Schwierigkeit des diskreten Logarithmusproblems) erklären.
- ☐ Ich kann den Man-in-the-Middle-Angriff auf Diffie-Hellman beschreiben und erläutern, wie digitale Signaturen (z.B. RSA) Authentizität sicherstellen.
- ☐ Ich kann eine Nachricht mit RSA asymmetrisch ver- und entschlüsseln, indem ich ein Beispiel mit kleinen Zahlen durchführe.
- ☐ Ich kann den Unterschied zwischen symmetrischen und asymmetrischen Verfahren (z.B. Vigenère/OTP vs. RSA/Diffie-Hellman) erklären.

Glossar

DHM Diffie-Hellman-Merkle. 34, 35

GGT Grösster Gemeinsamer Teiler. 38

OTP One-Time-Pad. 24–28, 36

RSA Rivest–Shamir–Adleman. 35, 37–40