SPL (Security)

Schuljahr 2023/2024

Anleitungen zu den Laborübungen

Wischounig Philipp

Abteilung für Wirtschaftsingenieure – Betriebsinformatik

HTBLVA Innsbruck

1 SPL zu SESD 1

Wir starten mit einem Brainstorming zu den Fragen:

- Wie könnte man verschlüsseln?
- Wann ist ein Verfahren sicher?
- Ist es praktikabel?

Chiffren lassen sich einteilen in **Stromchiffren (stream ciphers)**, bei denen z. B. Byte für Byte verschlüsselt wird, und **Blockchiffren (block ciphers)**, wo der Klartext in Blöcke gewisser Größe eingeteilt wird und zu Blöcken gleicher Größe verschlüsselt wird.

Grundlegende Mechanismen, um den Klartext (umkehrbar!) zu verschleiern, sind

. . . . . .

. . . . . .
eventuell erweitert um Einfügung von Sinnlosem.[1]

Chiffre 1: Rückwärtslesen

Klartext:	`GEHEIMNIS`
Chiffretext:	

Es handelt sich um eine . . . . . chiffre. Sie hat nur einen Schlüssel, lässt sich also nicht variieren. Kennt Eve die Chiffre, kann sie den Klartext aus dem Geheimtext ableiten.

Chiffre 2: Skytale

Ein Text wird zeilenweise auf einen um einen Holzstab gewickelten Lederstreifen geschrieben (für eine Abbildung siehe [2]).

Klartext:	`GEHEIMNIS`
Chiffretext:	

[1] siehe auch https://de.wikipedia.org/wiki/Fleißnersche_Schablone
oder https://en.wikipedia.org/wiki/Grille_(cryptography)
oder https://science.orf.at/stories/3216697/

[2] https://en.wikipedia.org/wiki/Scytale

Es handelt sich um eine ..... chiffre. Sie hat als Schlüssel

.....
Kennt Eve die Chiffre,

.....

.....
Bekannt ist sie aus der Antike. Aufgrund ihrer offensichtlich geringen Sicherheit spekulieren manche Historiker, sie sei zur **Authentifikation** statt Verschlüsselung verwendet worden.

Chiffre 3: Atbash

Der Name leitet sich vom hebräischen Alphabet ab:

Alef Bet ... Shin Tav

Der Name erklärt sich daraus, dass Alef durch Tav, Bet durch Shin usw. ersetzt wird. Übertragen auf das lateinische Alphabet hieße das, A durch Z, B durch Y usw. zu ersetzen. (Bei dieser und weiteren Chiffren könnte Tabelle 1 hilfreich sein.)

Klartext:	`BIBEL`
Chiffretext:	

Klartext:	
Chiffretext:	`SVYIVD`

Diese „Chiffrierung“ kommt teilweise in der Bibel vor (z. B. wird für „Babel“ der Begriff „Scheschach“ verwendet), vermutlich auch, um ihr eine geheimnisvolle Aura zu verleihen.

Es handelt sich um eine .....chiffre mit nur einem Schlüssel.

Zeichen	(dt.) rel. Häufigkeit	Nummer	ASCII-Code	Zeichen	ASCII-Code	Zeichen	ASCII-Code
A	5,58 %	00	65	a	97	LF	10
B	1,96 %	01	66	b	98	CR	13
C	3,16 %	02	67	c	99	␣	32
D	4,98 %	03	68	d	100	,	44
E	16,93 %	04	69	e	101	.	46
F	1,49 %	05	70	f	102	0	48
G	3,02 %	06	71	g	103	1	49
H	4,98 %	07	72	h	104	2	50
I	8,02 %	08	73	i	105	3	51
J	0,24 %	09	74	j	106	4	52
K	1,32 %	10	75	k	107	5	53
L	3,60 %	11	76	l	108	6	54
M	2,55 %	12	77	m	109	7	55
N	10,53 %	13	78	n	110	8	56
O	2,24 %	14	79	o	111	9	57
P	0,67 %	15	80	p	112	:	58
Q	0,02 %	16	81	q	113	;	59
R	6,89 %	17	82	r	114	?	63
S	6,42 %	18	83	s	115		
T	5,79 %	19	84	t	116		
U	3,83 %	20	85	u	117		
V	0,84 %	21	86	v	118		
W	1,78 %	22	87	w	119		
X	0,05 %	23	88	x	120		
Y	0,05 %	24	89	y	121		
Z	1,21 %	25	90	z	122		

Tabelle 1: Die Stellung der (Groß-)buchstaben im Alphabet und die ASCII-Codes der Buchstaben, Ziffern und wichtigsten Sonderzeichen. Des Weiteren sind die relativen Häufigkeiten der Buchstaben laut https://www.sttmedia.de/buchstabenhaeufigkeit-deutsch angeführt. Es fehlen dabei Ä (0,54 %), Ö (0,30 %), Ü (0,65 %) und ß (0,37 %).

Chiffre 4: Caesar-Verschlüsselung

Jeder Buchstabe wird durch jenen ersetzt, der im Alphabet k Stellen später vorkommt.[3] Für $k = 3$ ergibt sich die folgende Substitutionstabelle:

A	$\rightarrow$	D
B	$\rightarrow$	E
C	$\rightarrow$	F
$\vdots$		$\vdots$
Z	$\rightarrow$	C

Tabelle 2: Substitutionstabellen für die Caesar-Verschlüsselung mit Schlüssel $k = 3$.

Die Caesar-Verschlüsselung ist eine ..... chiffre mit .....

Schlüsseln. Damit ist sie anfällig für ....., wenn Eve die Methode kennt.[4]

Aufgabenstellungen

Die nachfolgenden Übungen sind prinzipiell als Einzelarbeit zu erledigen. Zum Austausch verschlüsselter Texte wird ein Partner/eine Partnerin benötigt.

Aufgabe 1: (Atbash händisch 1)
Verschlüssle mit dem *Atbash-Verfahren* **händisch**, d. h. mit Papier und Stift und ohne Python-Skripte, einen kurzen Text (ca. sechs bis zehn Wörter lang).

Aufgabe 2: (Python-Skripte kennenlernen)
Mach dich mit dem Programm `Monoalphabetic_encrypt.py` vertraut. Überprüfe deine Lösung mit der Python-Ausgabe und schicke sie anschließend deinem Laborpartner/deiner Laborpartnerin.

Aufgabe 3: (Atbash händisch 2)
Entschlüssle den Text deines Partners/deiner Partnerin **händisch und** mit Hilfe des Skripts.

Aufgabe 4: (Caesar händisch)
Verschlüssle einen anderen Text (ein mittellanger Satz) **händisch** mit Hilfe des *Caesar-Verfahrens*, überprüfe deine Lösung mit der Python-Ausgabe und schicke sie anschließend an deine Partnerin/deinen Partner. **Den Schlüssel schickst du nicht mit.**

[3] https://en.wikipedia.org/wiki/Caesar_cipher
[4] Trotzdem ist sie in Verwendung, siehe https://de.wikipedia.org/wiki/Pizzino.

Aufgabe 5: (Caesar knacken)
Entschlüssle den erhaltenen Geheimtext mit Hilfe des Python-Skripts. Das sollte machbar sein, auch ohne den Schlüssel zu kennen.

Dieser Angriff heißt .....

Ist er händisch realistisch durchführbar? Wie viele Buchstaben des Chiffretextes muss man zum Testen des Schlüssels untersuchen?(Muss man den gesamten Text lesen, um festzustellen, ob der Schlüssel passt?)

.....

.....

.....

Aufgabe 6: (Unizitätslänge)
Informiere dich über den Begriff der *Unizitätslänge*.

.....

.....

.....

.....

.....

.....

.....

.....

.....

Aufgabe 7: (Binärzahlen – Addition)
Addiere die folgenden Binärzahlen:

10101011 00101101 11010100	
10111111 01110101 11011001	$\oplus_{26}$

Wie viele Rechenschritte sind dafür nötig? .....
Angenommen, die zu addierenden Zahlen wären noch deutlich länger, aber du hättest deine KlassenkameradInnen, die dir helfen würden: Wären sie eine Hilfe? Könnte man die Addition so beschleunigen?

.....

.....

Aufgabe 8: (Binärzahlen – XOR)
Wende auf die folgenden Binärzahlen das bitweise „exklusive Oder“ an:

10101011 00101101 11010100	
10111111 01110101 11011001	XOR

Wie viele Rechenschritte sind dafür nötig? .....
Angenommen, die Zahlen wären noch deutlich länger, aber du hättest deine KlassenkameradInnen, die dir helfen würden: Wären sie eine Hilfe? Könnte man die Operation des bitweisen XOR so beschleunigen?

.....

.....

.....
Vergleiche noch einmal die erste Zahl und die Ergebnis-Zahl. Worin unterscheiden sie sich? **Wo** unterscheiden sie sich?

.....

.....

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Aufgabe 9: (Binärzahlen – Kryptographischer Vergleich von Addition und XOR)
Überlege, inwiefern sich Addition und XOR unterscheiden bzw. ähneln im Hinblick auf ihren Einsatz zur Verschlüsselung. Überlege beispielsweise, was es für das Caesar-Verfahren bedeuten würde, wenn man die Buchstaben nicht mit dem Schlüssel addieren würde sondern „ver-XOR-en“.

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Aufgabe 10: (XOR in Python)
Probiere die bitweise XOR-Operation in Python aus.

Kommandozeile: `4^12 = 8`

Erklärung: 4 XOR 12 = `00000100` XOR `00001100` = `00001000`

Wähle mehrere Beispiele selbst, sage das Ergebnis voraus und überprüfe die Voraussage mit der Python-Ausgabe.

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Chiffre 5: Allgemeine monoalphabetische Substitution

Als Verallgemeinerung des Caesar-Verfahrens werden hier nicht alle Buchstaben des Alphabets „gleich viel weiter geschoben“ sondern durch einen beliebigen anderen Buchstaben ersetzt. Weil diese Substitutionstabelle (das Substitutions„alphabet“) immer gleich bleibt, ist die Substitution *monoalphabetisch*.

A	→	J	oder	A	→	G
B	→	X		B	→	W
C	→	E		C	→	M
D	→	D		D	→	S
⋮		⋮		⋮		⋮
Z	→	I		Z	→	U

Tabelle 3: Beispielhafte Substitutionstabellen („Schlüssel“) für die monoalphabetische Substitution

Aufgabe 11: (Monoalphabetische Substitution – vereinfachtes Verschlüsseln)
Verschlüssle einen kurzen Text (ca. drei mittellange Sätze) mit einer *allgemeinen monoalphabetischen Substitution*. Vertausche dabei aber nur sieben halbwegs wichtige (also eher häufig vorkommende) Buchstaben, um das Brechen der Verschlüsselung zu vereinfachen. Die restlichen Buchstaben werden durch sich selbst ersetzt, müssen aber in den Boxen ausgewählt werden (U → u), damit es Kleinbuchstaben werden. Den erhaltenen Geheimtext schickst du deinem Partner/deiner Partnerin.

Aufgabe 12: (Monoalphabetische Substitution – Analyse im vereinfachten Fall)
Versuche durch Analysieren des Geheimtextes und geschicktes Ausprobieren auf den Klartext zu kommen. Wäre es auch gelungen, wenn alle 26 Buchstaben an der Substitution teilgenommen hätten?

. . . . . .

. . . . . .

. . . . . .

Aufgabe 13: (Monoalphabetische Substitution – vereinfacht – Anzahl der Schlüssel)
Berechne die Größe des Schlüsselraumes, d. h. die Anzahl an Möglichkeiten, von den 26 Buchstaben zuerst sieben auszuwählen und diese dann untereinander zu vertauschen. (Du darfst dabei erlauben, dass nicht alle sieben gewählten Buchstaben Plätze tauschen, also dass unter Umständen weniger als sieben Buchstaben ersetzt werden.)

.....

Aufgabe 14: (Monoalphabetische Substitution – Anzahl der Schlüssel)
Berechne die Größe des Schlüsselraumes, d. h. die Anzahl an Möglichkeiten, für die allgemeine monoalphabetische Substitution. (Natürlich schränken wir uns dabei **nicht** auf fixpunktfreie Permutationen ein, erlauben also, dass Buchstaben durch sich selbst ersetzt werden können. Eve wüsste sonst, dass bei jedem `E` im Chiffretext kein `E` im Klartext sein darf, was ihr das Leben deutlich vereinfacht! Rein kombinatorisch würde die Anzahl der Schlüssel so ungefähr um den Faktor $\frac{1}{e}$ reduziert.)

.....

Aufgabe 15:
Wir üben das Umrechnen von Potenzen zu den Basen 10 und 2.

(a) Rechne die Anzahl der Bankomatkarten-PINs in Potenzen um:

$$10000 = 10^x = 2^y$$

Nebenrechnung: .....
In dieser Gleichung muss

$x =$ ..... und $y =$ ..... sein.

(b) Gib die Formel an, mit der man von der Anzahl der Möglichkeiten auf den Exponenten zur Basis 2 kommt:

.....

(c) Gib die Formel an, mit der man vom Exponenten zur Basis 10 auf den zur Basis 2 kommt:

.....

(d) Finde einfache Hochzahlen (kopfrechnungstaugliche) für die Basen 10 und 2, um sich einen einfachen Umrechnungsfaktor merken zu können:

.....

Aufgabe 16: (Bitzahl zur Zahlgröße)
Leite einen formelmäßigen Zusammenhang zwischen einer Zahl und der Anzahl ihrer Dezimalziffern her. Betrachte dazu beispielsweise die folgenden Zahlen:

$$1, 10, 100, 1000, 10\,000, 100\,000\,000$$

und stelle eine Formel auf, die in diesen Spezialfällen die Anzahl der Dezimalziffern von n richtig berechnet:

$z_{10}(n) =$ .....
Überprüfe deine Formel für die folgenden Zahlen:

$$6, 60, 600, 6000, 60\,000\,000$$

Passe deine Formel so an, dass sie auch in diesem Fall stimmt:

$z_{10}(n) =$ .....
Passe deine Formel so an, dass sie die Anzahl der Binärziffern einer Zahl n berechnet:

$z_2(n) =$ .....
Wir interessieren uns vor allem für das asymptotische Verhalten der Binärstellen. Formuliere das Wesentliche dieser Formel in Worten:

.....

.....
Wenn eine Zahl verdoppelt wird, bedeutet das für ihre Binärziffern, dass

.....
Wenn sich die Anzahl der Binärziffern verdoppelt, wird die Zahl

.....

.....

Aufgabe 17:
Wir lernen Bit als „Einheit“ für den Informationsgehalt kennen.

(a) Bei einem Spiel denkt A an einen Schüler/eine Schülerin deiner Klasse und du musst herausfinden welche/n. Du darfst dazu ja/nein-Fragen stellen. Angenommen, du stellst die Fragen geschickt, wie viele musst du stellen, um auf die richtige Antwort zu kommen?

.....

. . . . . .

(b) Das Spiel wird jetzt mit allen Menschen auf der Erde (9 Milliarden) gespielt. Wie viele Fragen sind nötig? Schätze zuerst und rechne danach!

. . . . . .

Jede Antwort auf eine dieser ja/nein-Fragen kann als 1/0, also ein Bit, gespeichert werden.

Aufgabe 18:
Der Informationsgehalt von Variablen und Passwörtern wird üblicherweise in Bit angegeben.

(a) Berechne die Anzahl aller möglichen Zustände, die eine Variable vom Typ *Byte* annehmen kann:

. . . . . .

(b) Berechne die Anzahl aller möglichen Zustände, die eine Variable vom Typ *short int* (16 Bit) annehmen kann:

. . . . . .

(c) Berechne die Anzahl aller möglichen Zustände, die eine Variable vom Typ *int* (32 Bit) annehmen kann:

. . . . . .

(d) Berechne die Anzahl aller möglichen Passwörter mit 8 Zeichen, die nur aus Kleinbuchstaben bestehen (als Gleitkommazahl und als Potenz zur Basis 2):

. . . . . .

Wie aus den Teilaufgaben davor ersichtlich wird, kann man den Exponenten zur Basis 2 als die Anzahl an Bit interpretieren, die man benötigt, um eine Variable zu speichern, die entsprechend viele verschiedene Zustände speichern kann. Möchte man also eine Liste von Passwörtern nach obiger Richtlinie speichern, müsste man (platzsparend) wie viele Bit pro Passwort reservieren?

. . . . . .

(e) Berechne die Anzahl aller möglichen Passwörter mit 8 Zeichen, die nur aus Kleinbuchstaben, Großbuchstaben, Ziffern und zehn erlaubten Sonderzeichen bestehen (als Gleitkommazahl und als Potenz zur Basis 2):

. . . . . .

(f) Berechne die Anzahl aller möglichen Passwörter mit 16 Zeichen, die nur aus Kleinbuchstaben bestehen (als Gleitkommazahl und als Potenz zur Basis 2):

. . . . . .

Formuliere eine Schlussfolgerung aus den letzten drei Teilaufgaben, dabei darf auch Wissen aus dem Mathematikunterricht einfließen:

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Wie verlangen trotzdem Passwortrichtlinien meistens die Verwendung von Groß- und Kleinbuchstaben, Ziffern und Sondernzeichen?

. . . . . .

. . . . . .

(g) Berechne die Anzahl aller möglichen „Passwörter“/Schlüssel, die aus 8 Byte bestehen. (Die 8 Byte könnte man sich wie acht ASCII-Zeichen vorstellen, also z. B. acht ASCII-codierte Buchstaben, Ziffern oder Sonderzeichen. Es dürfen aber auch nicht-druckbare ASCII-Zeichen enthalten sein wie beispielsweise `LF` (line feed), `CR` (carriage return), `HT` (horizontal tabulation) uvm.)

. . . . . .

Jemand möchte eine Strichliste führen, wie oft welcher 8 Byte-Schlüssel von den BenutzerInnen verwendet wird. Dazu will er zuerst eine Liste aller dieser Schlüssel anlegen und später Striche daneben machen. Wir denken also an eine Textdatei, die folgendes enthält:

```
aaaaaaaa
aaaaaaab
aaaaaaac
   ⋮
aaaaaazz
aaaaaaAa
   ⋮
4!38andW
   ⋮
```

Schätze (ohne zu rechnen), wie groß diese Textdatei wird (Mail-Anhang, USB-Stick, Festplatte, Cloud etc.):

. . . . . .

Berechne die Größe (die Zeilenwechsel kannst du ignorieren):

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Vergleiche auch noch einmal mit der Anzahl an Schlüsseln für die monoalphabetische Substitution.

Obwohl es für die allgemeine monoalphabetische Substitution eine sehr große Anzahl an Schlüsseln gibt, ist sie bei ausreichend langen Klartexten aus natürlichen Sprachen leicht zu brechen (Unizitätslänge von ca. 26).

Aufgabe 19: (Monoalphabetische Substitution – Analyse)
Knacke den per *monoalphabetischer Substitution* verschlüsselten Geheimtext, den du von der Lehrperson erhältst.
War es ab einem bestimmten Zeitpunkt, zu dem der Inhalt klar wurde, leichter, den Rest zu knacken (Stichwort: *Known-ciphertext-Angriff*)?

. . . . . .

Kryptoanalyse – Angriffe

Selbst ein **Ciphertext-Only-Angriff** ist sehr aussichtsreich gegen Chiffren wie

. . . . . .

Manche Chiffren muss man lediglich **kennen**, um Chiffretext entziffern zu können. Solche Chiffren sind viel zu schwach. Andere haben so kleine **Schlüsselräume**, dass sie mit **Brute-Force-Angriffen** leicht zu brechen sind, z. B.

. . . . . .

. . . . . .

Dabei nutzt man die **Redundanz natürlicher Sprachen**: Nicht jede Buchstabenkombination ist sinnvoll!

- Vergleiche dazu „aber ich“ mit „eisn lai“.
- Stelle eine Vermutung auf, welcher Klartextbuchstabe nach „Ich möch“ kommt.

Vielleicht ist das Erraten des Schlüssels zu verschlüsselten Bild- oder Audiodaten schon schwieriger. **Für verschlüsselte Zufallszahlen ist es unmöglich.**

Kennen Angreifer schon etwas über den Plaintext, ist die Kryptoanalyse gleich einfacher (siehe Beispiel „Ich möch“, vergleiche mit den Erfahrungen beim Knacken der monoalphabetischen Substitution). Einem **Known-Plaintext-Angriff**, bei dem sogar die Position eines Klartextwortes bekannt ist, ist die monoalphabetische Substitution schutzlos ausgeliefert! Viele Buchstabenzuordnungen (vor allem die häufigen!) sind dann schon bekannt.

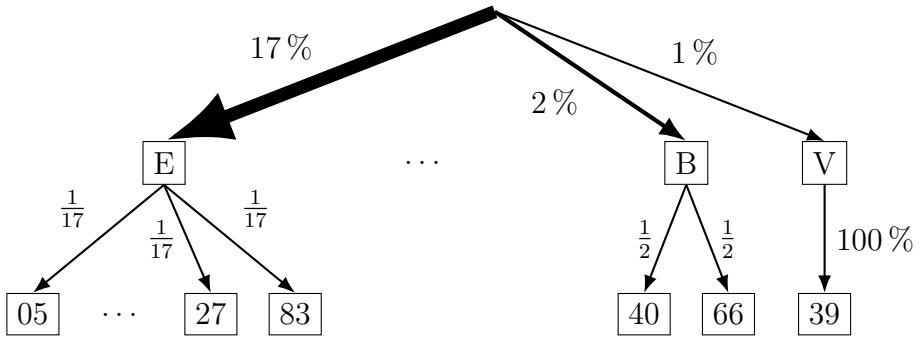
Aber auch rein die Information, in welcher Sprache der Klartext geschrieben ist, reicht offensichtlich aus, um die monoalphabetische Substitution leicht mittels **Häufigkeitsanalyse** zu knacken. Die beeindruckende Größe des Schlüsselraumes stellt sich als nutzlos heraus!

Ein großer Schlüsselraum ist **notwendig** für eine sichere Verschlüsselung aber **nicht hinreichend**!

Zur Abwehr der Häufigkeitsanalyse gilt es also, die Häufigkeitsverteilung abzuflachen. Einige Möglichkeiten hierfür sind die Folgenden:

Chiffre 6: Homophone Chiffre

Man ersetzt die 26 Buchstaben des Klartextes durch die 100 zweistelligen Zahlen. Für häufigere Buchstaben stehen mehrere Zahlen zur Verfügung, nämlich so viele wie der relativen Häufigkeit dieses Buchstabens in Prozent entspricht. Bei jedem Auftreten des Buchstabens beim Verschlüsseln wird zufällig eine dieser Zahlen gewählt wird.



Beispielsweise tritt in deutschen Texten das B doppelt so häufig auf wie das V. In der Hälfte der Fälle, wenn B zu verschlüsseln ist, lautet der Chiffretext 40, in der anderen Hälfte der Fälle 66. Das V wird jedes Mal als 39 chiffriert. Damit treten im Chiffretext die Zahlen 39, 40 und 66 annähernd gleich oft auf und eine Häufigkeitsanalyse wird verhindert.

Allerdings lässt sich auch diese Chiffre über Buchstaben*kombinationen* angreifen. Noch stärker ist ein Known-Plaintext-Angriff, wenn bereits ein paar Wörter erraten wurden.

Chiffre 7: Playfair-Chiffre

Diese Verschlüsselung versucht nicht, die Häufigkeitsstatistik aufzubrechen aber zu erschweren. Es werden keine Buchstaben durch andere substituiert sondern Buchstaben**paare**. Da es mehr Kombinationen aus zwei Buchstaben gibt als Buchstaben, nämlich

.....

wiederholen sich Paare auch seltener als Buchstaben.[5] Dadurch benötigt ein Angreifer viel mehr Chiffretext, um eine brauchbare Statistik zu erstellen. Des Weiteren hilft ihm die korrekte Zuordnung eines Paares weniger weiter.

Das Playfair-Verfahren ist eine .....alphabetische Substitutionschiffre. Sie wurde im Krimkrieg (1853-1856) und noch im Ersten Weltkrieg (1914-1918) verwendet, dort aber gebrochen.

Der Schlüsselraum war ja mit einzelnen Buchstaben schon riesig, hier wäre er noch viel größer. Um sich die Schlüssel irgendwie aufschreiben zu können, wurden stattdessen Playfair-Quadrate verwendet.[6] Damit ist sie aber viel schwächer als die allgemeine bigraphische Substitution. **Das Verfahren dient also der Vereinfachung, um den Schlüssel merkbar und die Bedienung leichter zu machen!**

Natürlich ließe sich das Verfahren auf Trigramme erweitern.

Eine weitere Möglichkeit ist, statt Bigrammen weiterhin Monogramme zu verwenden aber eine **polyalphabetische Substitution** durchzuführen, d. h. man verwendet nicht nur ein Substitutionsalphabet, sondern mehrere, je nach Position des Klartextbuchstabens. Das ist beispielsweise beim **Vigenère-Verfahren** der Fall.

Wir fassen zusammen:

- Selbst leicht knackbare Chiffren wie die monoalphabetische Substitution haben riesige Schlüsselräume, die selbst heutige Computer kaum mit Brute-Force durchsuchen können. Einen solchen Schlüssel zu speichern hieße im Wesentlichen, die Substitutionstabelle (wie in Tabelle 3) zu speichern (eigentlich reicht die rechte Spalte).

- Bei Verwendung von Bigrammen müsste man schon eine Tabelle mit 676 Einträge speichern.

- Blockchiffren aus den 1970ern verwenden 64 Bit-Blöcke. Im allgemeinen Fall müsste man die Zuordnung von jedem 64 Bit-Klartext-Block auf den zugehörigen 64 Bit-Chiffre-Block als Schlüssel speichern, siehe Abbildung 1.

  Das entspricht übrigens der Situation von Teilaufgabe (g) von Aufgabe 18. Ich könnte also den Schlüssel gar nicht speichern, so viele Möglichkeiten der Zuordnung gibt es.

  Andererseits ist diese unüberschaubare Menge an Zuordnungsmöglichkeiten nötig, um die Kryptoanalyse zu verhindern.

[5] siehe https://www.cryptool.org/de/cto/frequency-analysis
[6] siehe https://de.wikipedia.org/wiki/Playfair

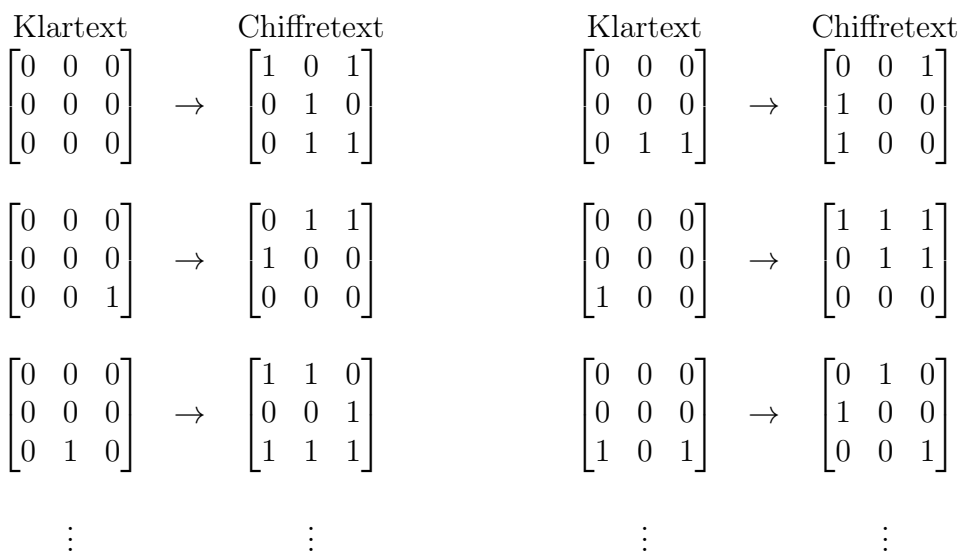


Abbildung 1: Eine monoalphabetische Substitution von Blöcken. Dargestellt sind 9 Bit-Blöcke. Auch diese Tabelle hier hätte 2^9 „Zeilen“, die je aus solchen Blöcken bestehen.

Die Antwort auf dieses Dilemma ist eben die Chiffre: Sie verwendet nur einen Teil der Zuordnungsmöglichkeiten, nämlich einen der mit (handhabbaren) Schlüsseln erzeugbar ist.

Das hat aber zur Folge, dass aus einem Klartext manche Chiffretexte nicht entstehen können, für keinen Schlüssel. Umgekehrt können aus einem Chiffretext manche Klartexte nicht entstehen, egal mit welchem Schlüssel. Das wiederum ermöglicht erst (zumindest theoretisch) Brute-Force-Angriffe: Finde ich einen Schlüssel, der zu einem plausiblen Klartext führt, ist er höchstwahrscheinlich richtig, weil fast alle Klartexte gar nicht zu diesem Chiffretext führen können.

Dass eine Chiffre aber nie alle Möglichkeiten ausnützt (das gilt zumindest für die meisten), hat aber auch zur Folge, dass durch **mehrfache Anwendung** der Chiffre vielleicht neue Möglichkeiten dazukommen, die durch Einfachanwendung nicht erreichbar gewesen sind.

Aufgabe 20:
Begründe, ob die mehrfache Anwendung der Caesar-Verschlüsselung oder der monoalphabetischen Substitution einen Zusatznutzen liefert.

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Chiffre 8: Permutationen

- Die Chiffre tauscht auf Blöcken der Länge 10 zwei Buchstaben aus:

Schlüssel:	(2,6)									
Klartext:	W	I	S	C	H	O	U	N	I	G
Chiffretext:	W	I	U	C	H	O	S	N	I	G

  Diese Chiffre hat . . . . . Schlüssel.

  Mehrfache Anwendung dieser Chiffre liefert neue Möglichkeiten! Zum Beispiel wäre das Folgende mit nur einfacher Anwendung nicht möglich gewesen:

Schlüssel:	(4,8) ∘ (2,6)									
Klartext:	W	I	S	C	H	O	U	N	I	G
	W	I	U	C	H	O	S	N	I	G
Chiffretext:	W	I	U	C	I	O	S	N	H	G

- Man könnte die Chiffre betrachten, die auf Blöcken der Länge 10 beliebige Permutationen zulässt. Sie hätte . . . . . Schlüssel. Bei ihr hätte Mehrfachanwendung

  . . . . . .

- Transpositionschiffren ändern die Buchstaben nicht und damit auch nicht ihre Häufigkeiten. Kennt eine Angreiferin die üblichen Buchstabenhäufigkeiten, kann sie die Sprache, in der der Klartext geschrieben wurde, leicht feststellen. Schlimmer noch: Vielleicht erfährt sie etwas über den Inhalt, wenn manche Buchstaben ungewöhnlich oft vorkommen.

- Da die Häufigkeiten sowieso nicht verändert wurden, ist eine Häufigkeitsanalyse als Angriff offensichtlich zwecklos.

- Trotzdem ist die Permutationschiffre leichter zu knacken als es auf den ersten Blick erscheint. Man tauscht dazu Buchstaben herum und analysiert die Häufigkeiten von Bigrammen (EN, ER, CH) und Trigrammen (ICH, EIN, UND). Es lassen sich viele Permutationen sehr schnell ausschließen, weil beispielsweise JH oder VV selten vorkommen, auf Q immer U folgt, ein C fast immer vor H oder K steht.

- Durchaus sinnvoll ist hingegen die Mehrfachanwendung auf Blöcken unterschiedlicher Länge, wenn die Längen darüber hinaus teilerfremd sind. Dann hat man ein durchaus solides Verfahren!

Der Kryptologe Klaus Schmeh stellte 2007 die Challenge, seinen Chiffretext zu knacken, der zweimal „verwürfelt“ wurde. Er benutzte zwei unterschiedlich lange Schlüsselwörter mit je über 20 Buchstaben. Er bekam erst 2013 eine Antwort.

Laut ihm zählt der „Doppelwürfel“ zu den besten Verfahren, für die kein Computer nötig ist. Es sei im Kalten Krieg beliebt gewesen, wo Agenten besser nicht mit dem seitenlangen Schlüssel (vgl. One-Time-Pad) oder gar einer Chiffriermaschine unterwegs waren.

- Transpositionschiffren haben eine lange Tradition als Rätsel (Anagramme) und auch zur Verschlüsselung in Kombination mit Authentifizierung:

  So entdeckte Galileo Galilei 1610 mit seinem Fernrohr, dass die Venus Phasen ähnlich wie der Mond aufweist, wollte das aber vorerst nicht veröffentlichen. Um später aber beweisen zu können, dass er das schon vor etwaigen Konkurrenten wusste, schrieb er Johannes Kepler „Haec immatura a me iam frustra leguntur oy“. Das ist ein Anagramm von „Cynthiae figuras aemulatur Mater Amorum“, „Die Gestalten Cynthias (Cynthia ist ein anderer Name für die Mondgöttin Artemis.) werden von der Mutter der Liebe nachgeahmt“.

Wir erwähnen noch ein paar Transpositionschiffren:

Chiffre 9: Rail fence

Ein *rail fence* ist ein Zaun mit Querbalken. Wir stellen uns vor, dass die Buchstaben des Klartextes im Zick-Zack zwischen den Querbalken diagonal nach unten und wieder nach oben laufen. Wenn der Abstand zwischen den Querbalken größer ist, passen mehr Buchstaben in eine Diagonale.

Wenn der Text zu Ende ist, werden **keine** Leerzeichen o. Ä. eingetragen, um eine „Schräglatte“ aufzufüllen. Das heißt, die Zeilen sind im Allgemeinen unterschiedlich lang.

Klartext:	Schafe ohne Schäfer brauchen einen Zaun.
Abstand 4:	S S E U E Z C E O C F R A C I A H F H E H E R H N N N U . A N A B E E N
Chiffretext:	S SEUEZCEO CFRAC I AHFHEHE RHNNNU.ANABEEN
Abstand 8:	S H N C C A E H S E H E . A F C I N F E E U N U E N R A E A H R N Z O B
Chiffretext:	SHNCCAE HSEHE.A FCINFEEUNUENRAEA H RNZOB

Chiffre 10: Redefence

Das Schlüsselwort wird links nach unten geschrieben und gibt die Anzahl der Zeilen vor. Der Klartext wird im Zick-Zack diagonal geschrieben und zeilenweise abgelesen. Die Reihenfolge, in der die Zeilen abgelesen werden, wird durch das Schlüsselwort vorgegeben, das links senkrecht steht. Die Buchstaben des Schlüsselwortes geben – in alphabetischer Reihenfolge abgelesen – vor, wann die Zeile an der Reihe ist.

Wenn der Text zu Ende ist, werden **keine** Leerzeichen o. Ä. eingetragen, um eine Diagonale aufzufüllen. Das heißt, die Zeilen sind im Allgemeinen unterschiedlich lang.

Klartext:	lasciate ogni speranza
Schlüssel: INFERNO	I 2: L I N 3: A N F 1: S G S E 0: C O P A R 6: I E Z N 4: A E R N O 5: T A
Chiffretext:	COPASGSLIAN AERNTAI EZ
Schlüssel: HELL	H 1: L T I A E 0: A A E N R N L 2: S I G S E Z L 3: C O P A
Chiffretext:	AAEN RNLTIASI GSEZCOPA

Problematisch ist, dass es im Codewort zu Buchstabenwiederholungen kommen muss und deshalb die Anzahl der Zeilentausch-Möglichkeiten nicht unbegrenzt wachsen kann. Außerdem ergeben viele Codewörter denselben Schlüssel (z. B. `CODE` und `ASEN`).

Chiffre 11: Rotation

Der Klartext wird von links nach rechts in n Spalten geschrieben. Diese Tabelle aus Zeichen des Klartextes wird um 90° oder 270° gedreht. Der Chiffretext wird „gewöhnlich“ von links nach rechts zeilenweise abgelesen.

Wenn der Text zu Ende ist, werden **keine** Leerzeichen o. Ä. eingetragen, um die Tabelle aufzufüllen. Das heißt, die Spalten (später Zeilen) sind im Allgemeinen unterschiedlich lang.

Klartext:	Österreich ist eine demokratische Republik. Ihr Recht geht vom Volk aus

Schlüssel: 7	`O E S T E R R`
	`E I C H   I S`
	`T   E I N E`
	`D E M O K R A`
	`T I S C H E`
	`R E P U B L I`
	`K .   I H R`
	`R E C H T   G`
	`E H T   V O M`
	`  V O L K   A`
	`U S`

Chiffretext:	mit 90° RS A I GMARIERELR O E NKHBHTVKTHIOCUIH L SCEMSP CTOEI EIE.EHVSOETDTRKRE U
Chiffretext:	mit 270° U ERKRTDTEOSVHE.EIE IEOTC PSMECSL HIUCOI HTKVTHBHKN E O RLEREIRAMG I A SR

Die Anzahl der Schlüssel ist offensichtlich überschaubar, weshalb ein Brute-Force-Angriff wohl erfolgversprechend wäre.

Chiffre 12: Spaltentausch

Das Schlüsselwort wird oben so angeschrieben, dass jeder seiner Buchstaben eine Spaltenüberschrift bildet. Der Klartext wird zeilenweise darunter geschrieben und spaltenweise abgelesen. Die Buchstaben des Schlüsselwortes geben – in alphabetischer Reihenfolge abgelesen – vor, wann die jeweilige Spalte an der Reihe ist.

Wenn der Text zu Ende ist, werden **keine** Leerzeichen o. Ä. eingetragen, um eine Zeile aufzufüllen. Das heißt, die Spalten sind im Allgemeinen unterschiedlich lang.

Wenn das Schlüsselwort einen Buchstaben öfters enthält, kommt in der Standardversion dieser Chiffre die erste Spalte mit diesem Buchstaben als erstes dran, dann die zweite usw.

In der Myszkowski-Variante wird mit im Schlüsselwort mehrfach auftretenden Buchstaben anders verfahren: Der Text nicht spaltenweise ausgelesen sondern in diesen Spalten **gemeinsam** zeilenweise von links nach rechts.

Klartext:	Freude schöner Götterfunken
Schlüssel: YOLO	Y O L O
alphabet. Reihenfolge:	3 1 0 2
☐ Use Myszkowski	--------
	F R E U
	D E S
	C H O E
	N E R
	G O E T
	T E R F
	U N K E
	N
Chiffretext:	E ORERKREHEOENUSE TFEFDCNGTUN
Schlüssel: YOLO	Y O L O
alphabet. Reihenfolge:	2 1 0 1
☒ Use Myszkowski	--------
Chiffretext:	E ORERKRUESHEE OTEFNEFDCNGTUN

Es gelten ähnliche Einschränkungen wie bei Redefence und Rotation.

Chiffre 13: Doppelter Spaltentausch

Diese Chiffre funktioniert gleich wie die Chiffre *Spaltentausch*. Allerdings gibt es zwei Schlüsselwörter, mit dem ersten wird *Spaltentausch* durchgeführt, das Ergebnis wird dann mit dem zweiten Schlüsselwort wieder einem *Spaltentausch* unterzogen.

Klartext:	HYPOTENUSE
Schlüssel: ECK	E C K
alphabet. Reihenfolge:	1 0 2

	H Y P
	O T E
	N U S
	E
Zwischentext:	YTUHONEPES
Schlüssel: DREI	D R E I
alphabet. Reihenfolge:	0 3 1 2

	Y T U H
	O N E P
	E S
Chiffretext:	YOEUEHPTNS

Da das spaltenweise abgelesene Ergebnis des ersten Spaltentauschs beim zweiten Durchgang zeilenweise geschrieben wird, bringt der zweite Durchgang durchaus viel Veränderung und einen Zusatz an Sicherheit.

Chiffre 14: Unterbrochener Spaltentausch – Kamm

Das Schlüsselwort wird oben so angeschrieben, dass jeder seiner Buchstaben eine Spaltenüberschrift bildet. Der Klartext wird zeilenweise darunter geschrieben, aber in der ersten Zeile nur bis zum alphabetisch ersten Buchstaben des Schlüsselwortes, in der zweiten Zeile bis zum alphabetisch zweiten Buchstaben des Schlüsselwortes usw. Der Chiffretext wird spaltenweise abgelesen, dabei gibt wieder die alphabetische Reihenfolge der Schlüsselbuchstaben die Reihenfolge der Spalten vor.

Wenn das Schlüsselwort einen Buchstaben öfters enthält, kommt die erste Spalte mit diesem Buchstaben als erstes dran, dann die zweite usw.

Klartext:	Wir betreten feuertrunken, Himmlische, dein Heiligtum.
Schlüssel: EUROPA alphabet. Reihenfolge:	E U R O P A 1 5 4 2 3 0 ---------- W I R B E T R E T E N F E U E R T R U N K E N , H I M M L I S C H E , D E I N H E I L I G T U M .
Chiffretext:	E LWTRNERNHII,ENIG. EENLHEMBU, EIRTFTEMCDHUIE RUKMS I T

Chiffre 15: Unterbrochener Spaltentausch – Numerischer Zugang

Das Schlüsselwort wird oben so angeschrieben, dass jeder seiner Buchstaben eine Spaltenüberschrift bildet. Der Klartext wird zeilenweise darunter geschrieben, enthält aber zusätzliche Leerzeichen.

Dazu wird die alphabetische Reihenfolge der Buchstaben eines zusätzlichen Unterbrechungs-Schlüsselwortes verwendet. Wenn der erste Buchstabe dieses Unterbrechungs-Schlüsselwortes der alphabetisch dritte Buchstabe im Unterbrechungs-Schlüsselwort ist, wird nach zwei Buchstaben Klartext ein Leerzeichen eingefügt. Wenn der nächste Buchstabe des Unterbrechungs-Schlüsselwortes an alphabetisch fünfter Stelle steht, wird nach vier weiteren Buchstaben Klartext ein Leerzeichen eingefügt usw.

Der Chiffretext wird spaltenweise abgelesen, dabei gibt die alphabetische Reihenfolge der Buchstaben des Schlüsselwortes die Reihenfolge der Spalten vor.

Wenn das Schlüsselwort einen Buchstaben öfters enthält, kommt die erste Spalte mit diesem Buchstaben als erstes dran, dann die zweite usw.

Klartext:	Bundesverfassungsgesetz
Schlüssel: PARLAMENT	P A R L A M E N T
alphabet. Reihenfolge:	6 0 7 3 1 4 2 5 8

	B U N D E S V
	E R F A S S U N
	G S G E S E
	T Z
Unterbrechungs-	
Schlüssel: AUSTRIA	A U S T R I A
Nach so vielen Buchstaben	0 6 4 5 3 2 1
wird ein Leerzeichen eingefügt.	
Chiffretext:	BRG DA SSSN GZESE U E UFSTVNE

Chiffre 16: ADFGVX (Spezialfall)

Die Buchstaben des Klartextes werden zuerst gemäß dieser Tabelle[7] durch Bigramme ersetzt, da sich die Buchstaben A, D, F, G, V und X besonders zum Morsen eignen:

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

Der so entstandene Zwischentext wird einem Spaltentausch unterzogen:

Das Schlüsselwort wird oben so angeschrieben, dass jeder seiner Buchstaben eine Spaltenüberschrift bildet. Der Zwischentext wird zeilenweise darunter geschrieben und spaltenweise abgelesen. Die Buchstaben des Schlüsselwortes geben – in alphabetischer Reihenfolge abgelesen – vor, wann die jeweilige Spalte an der Reihe ist.

Wenn der Text zu Ende ist, werden **keine** Leerzeichen o. Ä. eingetragen, um eine Zeile aufzufüllen. Das heißt, die Spalten sind im Allgemeinen unterschiedlich lang.

Wenn das Schlüsselwort einen Buchstaben öfters enthält, kommt die erste Spalte mit diesem Buchstaben als erstes dran, dann die zweite usw.

[7]Diese Tabelle entsteht durch den Tabellenschlüssel NACHTBOMMENWERPER. Er wird Buchstabe für Buchstabe in die Tabelle geschrieben, aber natürlich nur bei jeweils erstmaligem Auftreten. Dann folgen die nicht benutzten Buchstaben in alphabetischer Reihenfolge. Die Ziffer 1 kommt immer unmittelbar nach A, 2 nach B, 3 nach C usw.
siehe https://en.wikipedia.org/wiki/ADFGVX_cipher

Klartext:	Kaiser
Schlüssel: KOENIG	K O E N I G
alphabet. Reihenfolge:	3 5 0 4 2 1

	V G A D G X
	X A F A F G
Chiffretext:	AFXGGFVXDAGA

2 SPL zu SESD 2

Chiffre 17: Vigenère-Chiffre

Wir erinnern uns an das Caesar-Verfahren und die monoalphabetische Substitution. Sie waren beide sehr unsicher, da

.....

.....

Das Problem ist also, dass die Häufigkeitsstatistik der Buchstaben durch bloße Substitution mit nur einem Alphabet nicht verschwindet. Als Auswege wurden gefunden:

- .....

  .....

  .....

- .....

  .....

  .....

  .....

  .....

- .....

  .....

  .....

Die Vigenère-Chiffre ist einer der bekanntesten Vertreter der letzten Kategorie. Wir erinnern uns an das Caesar-Verfahren, bei dem Buchstaben um einen bestimmten Wert k, den Schlüssel, „verschoben“ werden. Stellt man sich die Buchstaben (wir verwenden hier in der Demonstration nur Großbuchstaben) als Zahlen vor ($A \to 0$, $B \to 1$,..., $Z \to 25$), dann lässt sich das Chiffretext-Zeichen c_i aus dem Klartext-Zeichen m_i über

.....

berechnen. (Den erhaltenen Zahlenwert ersetzt man dann wieder durch den zugehörigen Buchstaben.)

Beim Vigenère-Verfahren wird nun nicht eine „Verschiebungszahl“ k als Schlüssel verwendet sondern ein Schlüsselwort, das wiederholt unter den Klartext geschrieben wird. Die Buchstaben des Schlüsselwortes (interpretiert als Zahlen) sind nun der Schlüssel, mit dem der darüber stehende Klartext-Buchstabe „Caesar-verschlüsselt“ wird:

Als Beispiel betrachten wir die Verschlüsselung von $m = \texttt{EINGEHEIMNIS}$ mit Hilfe des Schlüssels $k = \texttt{HUND}$, der in diesem Fall dreifach wiederverwendet werden muss.

Klartext:	EINGEHEIMNIS	=	04	08	13	06	04	07	04	08	12	13	08	18	
Schlüssel:	HUNDHUNDHUND	=	07	20	13	03	07	20	13	03	07	20	13	03	$\oplus_{26}$
Chiffret.:	LCAJLBRLTHVV	=	11	02	00	09	11	01	17	11	19	07	21	21	

Die Buchstaben des Klartextes, die mit derselben Farbe markiert sind, werden mit demselben Schlüssel-Zeichen verschlüsselt. Sie unterliegen also nur einer Caesar-Verschlüsselung. Die Klartextzeichen mit einer anderen Farbe unterliegen einer **anderen** Caesar-Verschlüsselung, also einer mit anderem Schlüssel.

Den Vorteil dieses Verfahrens sieht man, wenn man den im Klartext dreimal vorkommenden Buchstaben `I` ansieht: Er wird dreimal unterschiedlich verschlüsselt, nämlich zu `C`, `L` und `V`.

Umgekehrt sind die letzten beiden Zeichen des Chiffretexts gleich (`V`). Diese Tatsache lässt sich für einen Angriff aber nicht nützen, weil die Häufung des `V` keiner Häufung im Klartext entspringt.

Grafisch kann man die Verwaschung der Buchstabenhäufigkeiten in Abbildung 2 nachvollziehen. Die Buchstabenhäufigkeiten der deutschen Sprache werden an allen Stellen, die mit `H` verschlüsselt werden, nicht verändert, außer dass sie „um 7 Buchstaben weiter rechts“ stehen. An jenen Stellen, an denen mit `U` verschlüsselt wird, hat man wieder die gleiche Statistik zu erwarten, bloß weitere 13 Buchstaben „später“. Durch die unterschiedlichen Verschiebungen werden lange Balken meist an anderen Stellen landen.

Addiert man nun die Häufigkeiten für die verschiedenen Schlüssel-Zeichen, ergibt sich eine Häufigkeitsverteilung, die deutlich flacher ist als die der deutschen Sprache. Ein Angriff auf die Buchstabenhäufigkeiten im Chiffretext ist damit deutlich schwerer.

Die Vigenère-Chiffre stammt aus dem 16. Jahrhundert und galt lange als unknackbar (*le chiffre indéchiffrable*). Da sie einfach anzuwenden ist (auch von Laien und ohne besondere Hilfsmittel), wurde sie beispielsweise auch noch im Amerikanischen Bürgerkrieg (1861-1865) eingesetzt. Allerdings wurden damals bereits regelmäßig Nachrichten von der Union geknackt – die Konföderation verwendete auch nur drei Schlüssel im ganzen Krieg.[8]

[8] siehe https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher, auch für Bilder

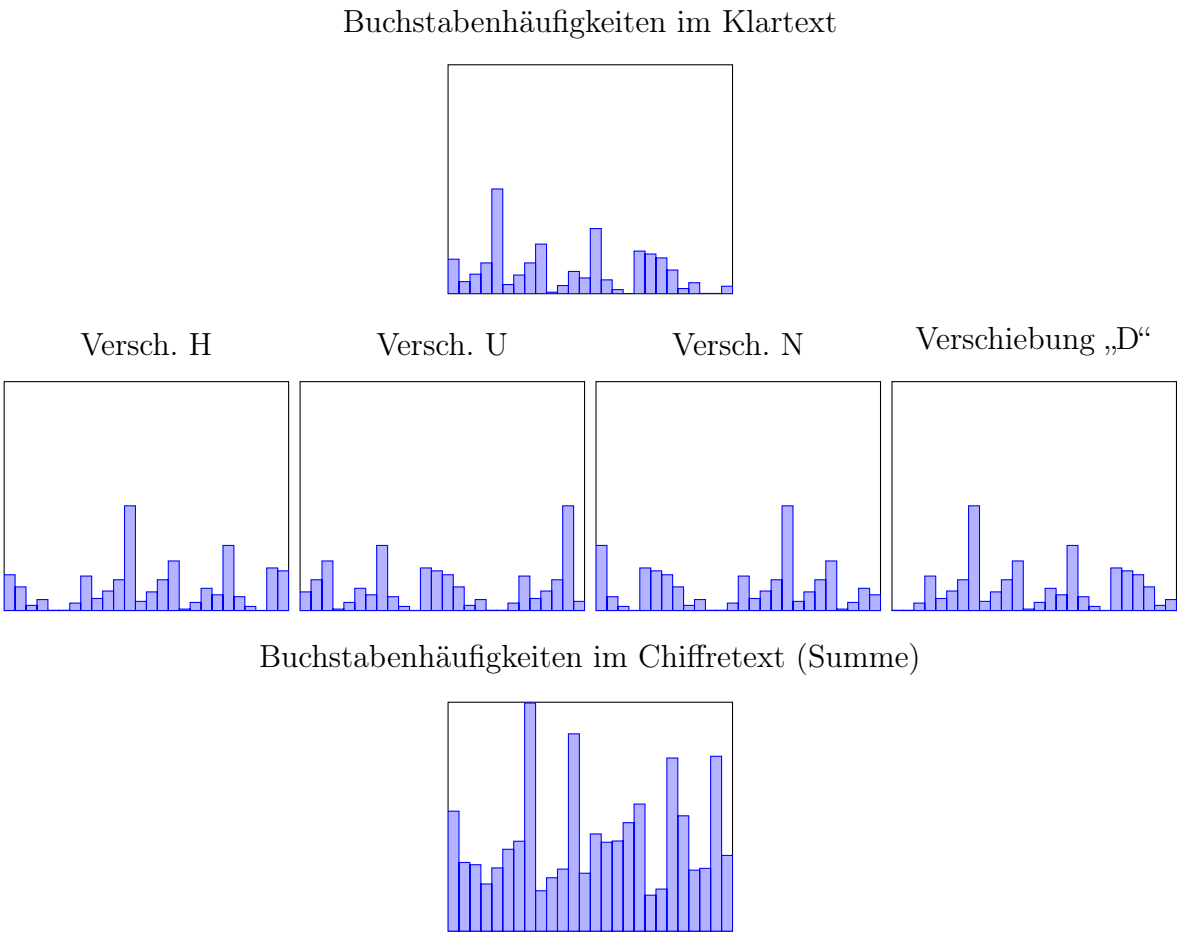


Abbildung 2: Im Diagramm KT sind die relativen Buchstabenhäufigkeiten für die deutsche Sprache von A bis Z dargestellt, so wie sie in einem durchschnittlichen Klartext auftreten würden.
In den folgenden vier Diagrammen sind die Häufigkeiten zu sehen, wie sie an allen Stellen, an denen mit H, U, N bzw. D verschlüsselt wird, zu erwarten sind. Es handelt sich also um dieselben Häufigkeiten, die lediglich gegeneinander verschoben sind.
Im letzten Diagramm ist die Summe dieser vier relativen Häufigkeiten dargestellt. Es gibt die Häufigkeitsverteilung im Chiffretext an. Man sieht, dass diese Häufigkeitsverteilung gleichmäßiger ist, weil lange Balken mit kurzen addiert werden und umgekehrt.

Aufgabenstellungen

Für die folgenden Übungen stehen Python-Skripts zur Verfügung. Das *Vigenère-Verfahren* ist so implementiert, dass der Buchstabe „A“ der Zahl 0 entspricht, „B“ der 1 usw., die entsprechenden Zahlen modulo 26 addiert werden und wieder zu Buchstaben zurück ersetzt werden. Wir arbeiten ausschließlich mit Großbuchstaben, Kleinbuchstaben werden zuerst durch Großbuchstaben ersetzt, alle anderen Zeichen bleiben unverändert. Beispiel:

$$\begin{aligned} \mathrm{A} + \mathrm{A} &= \mathrm{A} \\ \mathrm{B} + \mathrm{B} &= \mathrm{C} \\ \mathrm{X} + \mathrm{Y} &= \mathrm{V} \end{aligned}$$

Aufgabe 21: (Vigenère händisch 1)
Verschlüssle einen Text (ein kurzer Satz) **händisch** mit Hilfe des *Vigenère-Verfahrens*. Verwende dabei einen selbstgewählten Schlüssel. Überprüfe anschließende deine Lösung mit der Python-Ausgabe und schicke sie anschließend inklusive Schlüssel an deine Partnerin/deinen Partner.

Aufgabe 22: (Vigenère händisch 2)
Entschlüssle den erhaltenen Geheimtext **händisch**. Wie viel Arbeit hätte das Python-Skript erspart? Ist die Methode für die händische Verwendung vorstellbar? Für wie sicher hältst du das Verfahren (*le chiffrage indéchiffrable*)?

Kasiski-Angriff zur Bestimmung der Schlüssellänge

Nachdem der Vigenère-Chiffre mehrere Jahrhunderte nicht beizukommen war, veröffentlichte Friedrich Wilhelm Kasiski 1863 einen Ciphertext-Only-Angriff.

Bekanntlich werden häufig auftretende Buchstabenfolgen (`EN`, `ER` usw.) beim Vigenère-Verfahren oft verschieden verschlüsselt. Da der Schlüssel sich aber wiederholt, werden sie aber oft auch mit denselben Schlüssel-Zeichen verschlüsselt, erscheinen also auch im Chiffretext öfter. Deshalb wird der Chiffretext nach häufigen Digrammen (oder auch mehr als zwei Buchstaben, im Beispiel sind es vier) durchsucht:

Klartext:	`EINEKLEINEMICKEYMOUSEZIEHTSICHEINEHOSEAN`
Schlüssel:	`JEANSJEANSJEANSJEANSJEANSJEANSJEANSJEANS`
Chiffret.:	`NMNRCUIIAWVMCXWHQOHKNDIRZCWIPZNMNRZXWENF`

Der Klartextausschnitt `EINE` tritt dreimal auf, einmal wird er anders verschlüsselt, nämlich mit `EANS`, aber zweimal gleich (mit `JEAN`). Deswegen tritt im Chiffretext `NMNR` zweimal auf. Eine Angreiferin würde das bemerken und schlussfolgern:

- vielleicht ist es .....

  .....

- .....

  .....

  Dann muss die Schlüssellänge allerdings

  .....

  .....

Auf dieselbe Art sucht man noch nach weiteren häufigeren Schnipseln im Chiffretext und führt eine Strichliste, welche Teiler oft vorkommen. Da gleiche Chiffretext-Abschnitte auch durch Zufall entstehen können, werden auf dieser Liste auch „falsche“ Zahlen landen, aber manche Teiler stechen in ihrer Häufigkeit hervor, siehe beispielsweise Abbildung 3. Es lohnt sich, mit der Annahme weiter zu arbeiten, dass der Schlüssel diese Länge hat.

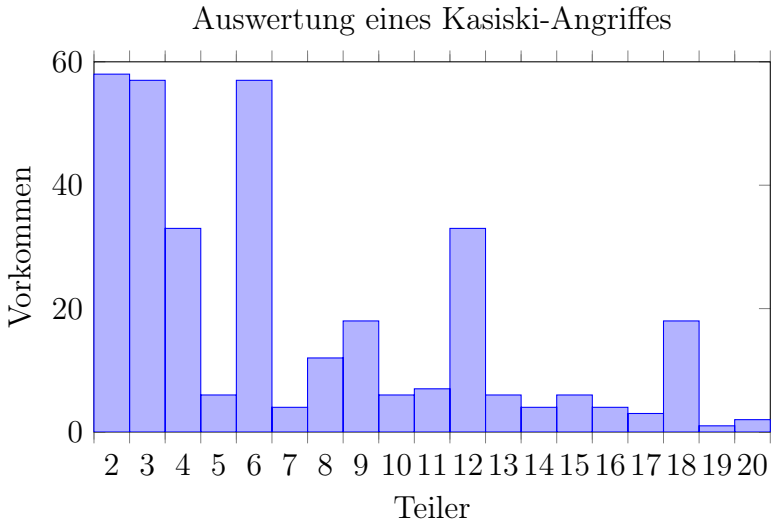


Abbildung 3: Zu sehen ist das Ergebnis eines Kasiski-Angriffs auf einen Beispieltext, wobei Trigramme ausgewertet wurden. Wurde ein Trigramm mehrfach gefunden, wurden alle Teiler ihres Abstandes gezählt. Im Säulendiagramm ist die Summe der Vorkommen dieser Teiler aufgezeichnet.
Da Trigramme auch „zufällig“ auftreten können, gibt es auch „Hintergrundrauschen“. Aber die systematisch auftretenden Teiler der wahrend Schlüssellänge kommen auf dieses Rauschen oben drauf und zeichnen sich klar erkennbar ab. Hier könnte man die Vermutung aufstellen, dass die Schlüssellänge 6 beträgt.

Sobald aber die Schlüssellänge l bekannt ist, müssen nur noch l Caesar-Verschlüsselungen gebrochen werden. Das geht schnell mit Hilfe von

.....

oder Koinzidenzindizes (siehe unten).

Aufgabe 23: (Vigenère knacken – graphisch)
Lass dir von der Lehrperson das Python-Skript zur Analyse des Vigenère-Verfahrens vorführen.
Knacke den per Vigenère-Verfahren verschlüsselten Chiffretext, den du von der Lehrperson erhältst.

Aufgabe 24: (Kasiski-Angriff – Schnipsellänge)
Überlege, wieso beim Kasiski-Angriff von Kombinationen aus zwei oder drei Buchstaben die Rede war? Welche Auswirkungen hätten längere, welche kürzere Stücke?
Spielt auch ein wenig mit CrypTool-Online.[9]

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Key elimination – ein Known-Plaintext-Angriff bei bekannter Schlüssellänge

Ist ein Teil des Klartextes bekannt oder lässt sich vermuten, gibt es auch noch einen anderen Angriff: Wenn die Angreiferin bereits die Schlüssellänge kennt oder vermutet (z. B. nach einem erfolgreichem Kasiski-Angriff) und darüber hinaus einen Teil des Klartextes weiß, kann sie den Known-Plaintext-Angriff *Key elimination* versuchen. Beachte dabei, dass sie zwar einen Teil kennen muss, der doppelt so lange wie der Schlüssel ist, aber die Position dieses Klartext-Stückes nicht bekannt sein muss.

Die Methode wird an einem Beispiel illustriert, siehe Tabelle 4. Der Chiffretext entsteht aus dem Klartext, indem der Schlüssel, der sich nach seiner Länge l wiederholt, dazuaddiert wird:

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

Wenn man zwei Chiffretext-Zeichen im Abstand der Schlüssellänge voneinander subtrahiert, ergibt sich

$$\begin{aligned} c_{i+l} - c_i &= (m_{i+l} + k_{(i+l \bmod l)}) \bmod 26 - (m_i + k_{(i \bmod l)}) \bmod 26 = \\ &= (m_{i+l} + k_{(i \bmod l)}) \bmod 26 - (m_i + k_{(i \bmod l)}) \bmod 26 \equiv (m_{i+l} - m_i) \pmod{26}, \end{aligned}$$

[9] https://www.cryptool.org/de/cto/frequency-analysis

Klartext:	A	L	L	E	M	E	I	N	E	E	N	T	C	H	E	N	S	C	H	W	I	M
Schlüssel:	S	E	E	S	E	E	S	E	E	S	E	E	S	E	E	S	E	E	S	E	E	S
Chiffret.:	S	P	P	W	Q	I	A	R	I	W	R	X	U	L	I	F	W	G	Z	A	M	E
Erkl.:	A+S	L+E	L+E	E+S	M+E	E+E	I+S	N+E	E+E	E+S	N+E	T+E	C+S	H+E	E+E	N+S	S+E	C+E	H+S	W+E	I+E	M+S

Klartext:		A	L	L	E	M	E	I	N	E	E	N	T	C	H	E	N	S	C	H		
Schlüssel:		S	E	E	S	E	E	S	E	E	S	E	E	S	E	E	S	E	E	S		
Chiffret. versch:		S	P	P	W	Q	I	A	R	I	W	R	X	U	L	I	F	W	G	Z		
Erkl.:		A+S	L+E	L+E	E+S	M+E	E+E	I+S	N+E	E+E	E+S	N+E	T+E	C+S	H+E	E+E	N+S	S+E	C+E	H+S		

KT – KTversch:		E-A	M-L	E-L	I-E	N-M	E-E	E-I	N-N	T-E	C-E	H-N	E-T	N-C	S-H	C-E	H-N	W-S	I-C	M-H		
CT – CTversch:		E-A	M-L	E-L	I-E	N-M	E-E	E-I	N-N	T-E	C-E	H-N	E-T	N-C	S-H	C-E	H-N	W-S	I-C	M-H		

Tabelle 4: Erläuterung der Key elimination an einem Beispiel. Die Angreiferin vermutet richtigerweise, dass irgendwo im Klartext ENTCHE vorkommt, was doppelt so lange wie der Schlüssel ist. Sie verschiebt nun ENTCHE um die Schlüssellänge nach rechts und subtrahiert es von ENTCHE. Sie erhält eine Differenz, die gleich lange wie der Schlüssel ist (blau).
Weil der Chiffretext nur aus der Summe von Klartext und Schlüssel besteht, das natürlich auch für den verschobenen Chiffretext gilt und der um die Schlüssellänge verschobene Schlüssel gerade wieder der Schlüssel ist, ergibt die Differenz aus Chiffretext und verschobenem Chiffretext dasselbe wie die Differenz aus Klartext und verschobenem Klartext. Die Schlüssel, die in Chiffretext und verschobenem Chiffretext hinzuaddiert sind, heben sich durch die Subtraktion auf!
Die Angreiferin bildet also die Differenz aus Chiffretext und verschobenem Chiffretext und sucht in ihr die blaue Klartext-Differenz. Wird sie fündig, kann sie erwarten, dass an dieser Stelle (!) tatsächlich der vermutete Klartextteil steht. Sie muss dann nur noch vom Chiffretext an dieser Stelle (rot) den Klartext an dieser Stelle (blau) abziehen und erhält den Schlüssel (grün). Mit diesem lässt sich der Rest entschlüsseln.

also dasselbe, wie wenn man die zwei zugehörigen Klartextzeichen voneinander subtrahiert.

Eine Angreiferin kann das jetzt nutzen, indem sie vom vermuteten Klartext seine um die Schlüssellänge verschobene Version subtrahiert. Diese Differenzbildung eliminiert den Schlüssel, der ja an den Stellen i und $i + l$ derselbe ist. Als nächstes zieht sie vom Chiffretext dessen verschobene Version ab. Findet sie anschließend die Klartext-Differenz in der Chiffretext-Differenz, weiß sie die Position des Klartext-Stückes. Wenn sie es noch vom zugehörigen Chiffretex-Stück abzieht, erhält sie den Schlüssel, mit dem sie leicht den Rest des Chiffretextes entschlüsseln kann.

- Die Methode hat den Nachteil, dass ein ausreichend langes Stück Klartext bekannt sein muss. Bei Sätzen, die als Schlüssel verwendet werden, ist das eine hohe Hürde.
- Ein Vorteil ist, dass die Methode auch funktioniert, wenn im Bekannten Klartext Lücken vorhanden sind (`um ??:?? Uhr`). Man findet vielleicht trotzdem die richtige Stelle des Klartextes und erfährt einen Teil des Schlüssels. Die Lücken dazwischen zu stopfen sollte leicht möglich sein.
- Ein weiterer Vorteil ist, dass zwar ein Stück Klartext benötigt wird, aber kein Klartext-Chiffretext-Paar, also die Position nicht bekannt sein muss.

Friedman-Schätzung – Koinzidenzindizes

William Friedman veröffentlichte 1922 einen statistischen Test, um herauszufinden, ob ein Chiffretext mono- oder polyalphabetisch verschlüsselt wurde. Die Ideen lassen sich auch auf zwei Arten zur Analyse der Vigenère-Chiffre einsetzen.

Hat man einen Klartext in einer natürlichen Sprache, treten die Buchstaben mit gewissen relativen Häufigkeiten $(f_A, f_B, \ldots, f_Z)$ auf, die von Sprache zu Sprache unterschiedlich sind. Wählt man nun aus einem vorliegenden Text zwei Buchstaben zufällig aus, dann ist die Wahrscheinlichkeit, dass die beiden Buchstaben gleich sind (siehe Abbildung 4):

$P(B_1 = B_2) =$ ....

Für Englisch ist dieser Wert in etwa 6,6 %, für Deutsch 7,6 %.

Ist ein Text monoalphabetisch verschlüsselt,

....

....

....

Hat man einen „Text“, in dem die Buchstaben alle gleich wahrscheinlich auftreten, ist der obige Ausdruck stattdessen

$P(B_1 = B_2) =$ .... $\approx 3{,}8\,\%$.

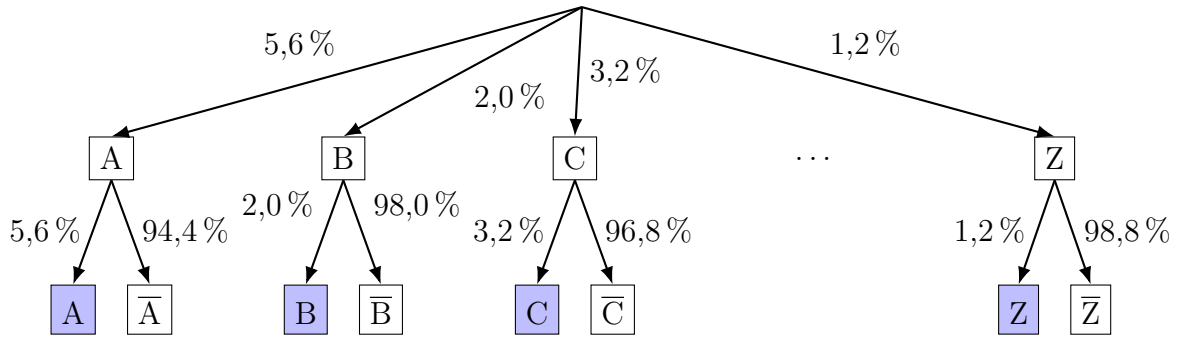


Abbildung 4: Zwei zufällig ausgewählte Buchstaben aus einem deutschen Text sind gleich (blaues Ereignis), wenn der erste `A` ist und der zweite `A` oder der erste `B` ist und der zweite `B` usw. Die Wahrscheinlichkeit einen jeweiligen Buchstaben zu ziehen ist die relative Häufigkeit des Buchstabens im Deutschen.

Liegt ein Vigenère-verschlüsselter Text vor, für den ein Schlüssel der Länge l verwendet wurde, dann gilt für zwei zufällig ausgewählte Zeichen das Folgende:

- Wurden beide Chiffretext-Zeichen mit demselben Schlüssel-Zeichen verschlüsselt, entspringen sie *derselben* monoalphabetischen Substitution. Die Wahrscheinleichkeit, dass sie dann gleich sind, ist folglich dieselbe wie für deutschen Text.
- Wurden beide mit unterschiedlichen Chiffretext-Zeichen verschlüsselt (und nehmen wir an dass im Schlüssel alle Buchstaben gleich wahrscheinlich und unabhängig sind!), haben sie miteinander nichts zu tun. Die Wahrscheinlichkeit, dass sie gleich sind, ist $\frac{1}{26} = 3{,}8\,\%$.

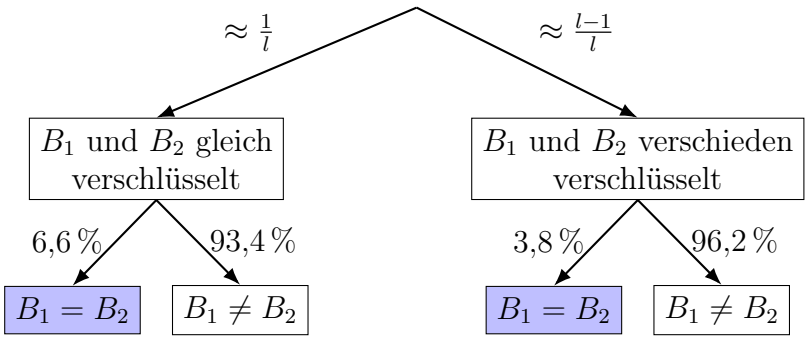


Abbildung 5: Wahrscheinlichkeitsbaum, dass zwei zufällig aus einem Vigenère-verschlüsselten Text ausgewählte Buchstaben gleich sind. Zuerst wird unterschieden, ob sie mit demselben Schlüssel-Zeichen verschlüsselt wurden. Falls ja, ist die Wahrscheinlichkeit für ihre Gleichheit dieselbe wie in der natürlichen Sprache. Falls nein, einfach $\frac{1}{26}$.

Die Wahrscheinlichkeit dafür, dass die zufällig gewählten Buchstaben mit demselben Schlüssel-Zeichen verschlüsselt wurden, ist aber gerade $\frac{1}{l}$. In Abbildung 5 ist der Wahr-

scheinlichkeitsbaum für die zufällige Wahl zweier Buchstaben aus einem Vigenère-verschlüsselten Text zu sehen. Die Wahrscheinlichkeit, dass sie gleich sind, ist:

$$P(B_1 = B_2) = \frac{1}{l} \cdot 6{,}6\,\% + \frac{l-1}{l} \cdot 3{,}8\,\%$$

Man sieht leicht ein, dass für umso längere Schlüssel es immer unwahrscheinlicher wird, dass zwei zufällig gewählte Chiffretext-Zeichen gleich verschlüsselt wurden. Dass sie gleich sind, liegt dann fast nur noch am Zufall. Der Wert für $P(B_1 = B_2)$ wird dann umso näher an 3,8 % liegen.

Der Koinzidenzindex $P(B_1 = B_2)$ des Textes lässt sich aus den Buchstabenhäufigkeiten des Chiffretextes berechnen und die Gleichung dann auf l umformen. So erhält man eine Schätzung für die Schlüssellänge.

Diese Methode ist somit eine Alternative zum Kasiski-Angriff, um die Schlüssellänge zu schätzen (oder eine Ergänzung, um eindeutigere Ergebnisse zu erhalten).

Allgemeiner lässt sich mit der Friedman-Methode feststellen, ob ein substitutionschiffrierter Text monoalphabetisch verschlüsselt ist (dann ist $l \approx 1$) oder polyalphabetisch.

Die Koinzidenzindizes sind aber ebenso nützlich, um bei bekannter Schlüssellänge die einzelnen Buchstaben des Schlüssels (die Caesar-Verschiebungen) herauszufinden.

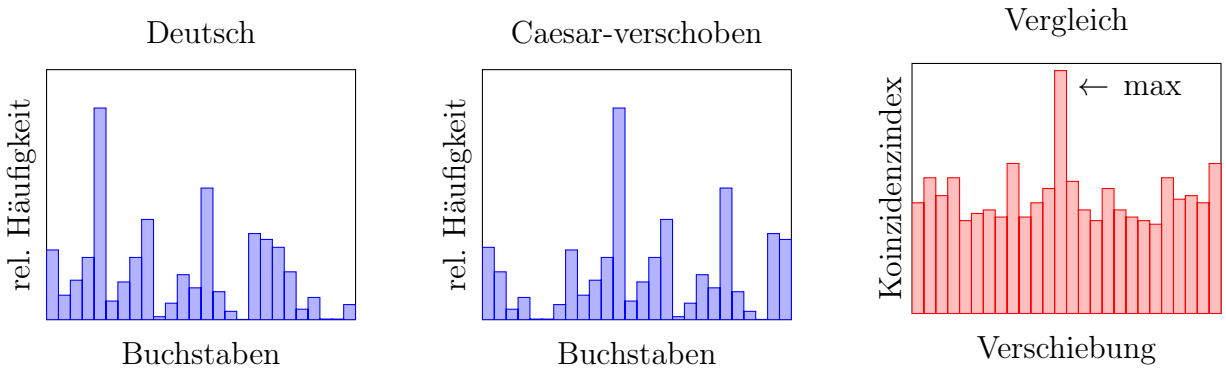


Abbildung 6: Multipliziert man jeweils die ersten Säulen, die zweiten usw. (also die relativen Häufigekeiten in deutschem Klartext und in Caesar-verschobenem Text) und addiert die Ergebnisse, kommt auf einen Koinzidenzindex. Dieser wird für einen Beispieltext aufgeschlüsselt nach Größe der Caesar-Verschiebung im Diagramm rechts aufgetragen. Wenn es „keine Caesar-Verschiebung“ gibt, also große Zahlen mit großen multipliziert werden und kleine mit kleinen, entsteht ein markantes Maximum.

Die Buchstabenhäufigkeiten aller gleich chiffrierten Zeichen entsprechen jenen der deutschen Sprache, sind aber verschoben, siehe Abbildung 6. Wählt man nun einen Buchstaben aus einem deutschen Text und einen aus dem chiffrierten, ist die Wahrscheinlichkeit, dass sie gleich sind, besonders dann hoch, wenn die Verschiebung null beträgt, also die hohen Säulen an denselben Stellen stehen.

Der Ausdruck

$$f_A \cdot h_{A+k} + f_B \cdot h_{B+k} + \ldots + f_Z \cdot h_{Z+k},$$

in dem die h_i für die relativen Häufigkeiten des i-ten Buchstabens im Chiffrat stehen, ist dann am größten, wenn $k = 0$. So lässt sich das Bestimmen der einzelnen Zeichen des Vigenère-Schlüssels automatisieren. Man erstellt eine Häufigkeitsstatistik unter allen Chiffretext-Zeichen, die gleich verschlüsselt wurden, und verschiebt sie so weit, bis sie am besten zur Häufigkeitsverteilung der deutschen Sprache passt.

Aufgabe 25:
Lass dir von der Lehrperson das Knacken einer Vigenère-Verschlüsselung mit Hilfe des AUTO-Knopfes im Python-Skript vorführen. Nach einem erfolgreichen Kasiski-Angriff lassen sich die korrekten „Ceasar-Verschiebungen“, also die Schlüsselwort-Buchstaben, leicht automatisiert herausfinden, weil die Berechnung von 26 Koinzidenzindizes und ihres Maximums mit dem Computer einfach ist.

Chiffre 18: Vernam-Verfahren

Das Problem der Wiederverwendung des Schlüsselwortes wird durch das Vernam-Verfahren gelöst. Es verwendet ein Schlüssel„wort“, das gleich lang ist wie der zu verschlüsselnde Klartext. Da das nicht merkbar ist, verwendet man typischerweise den Text eines leicht verfügbaren Buches (Telefonbuch, Bibel etc.).

Die Schlüsselraumgröße, um einen Klartext der Länge n zu verschlüsseln, ist

.....
also viel zu groß für einen Brute-Force-Angriff. Eher würde man selbstverständlich

.....

.....
Die einzigen Angriffsmöglichkeiten ergeben sich daraus, dass sowohl im Klartext als auch im Schlüssel manche Zeichen, Zeichenkombinationen und Muster häufiger auftreten.

Chiffre 19: Das One-Time-Pad

Um auch noch diese verbleibende Schwäche des Vernam-Verfahrens auszumerzen, ist die letzte Idee, eine **völlig zufällige** Kette aus **unabhängigen** und **gleichverteilten** Zeichen als Schlüssel zu verwenden. Die einzelnen Zeichen sind dann der Schlüssel für eine Caesar-Verschiebung für **ein** Zeichen des Klartextes. **Kein Schlüssel-Zeichen wird jemals wieder verwendet, daher der Name!**

Zur Demonstration werden nur die Großbuchstaben ($A \to 00$, $B \to 01$,..., $Z \to 25$) und die Addition modulo 26 verwendet.

Klartext:	ABEND	=	00	01	04	13	03	
Schlüssel:	EOVKL	=	04	14	21	10	11	$\oplus_{26}$
Chiffretext:	EPZXO	=	04	15	25	23	14	

Der Klartextbuchstabe N entspricht $m_4 = 13$. Zu diesem Klartextzeichen wird das vierte Zeichen des Schlüssels (K bzw. $k_4 = 10$) addiert, so dass das entsprechende Chiffretextzeichen $c_4 = (13 + 10) \bmod 26 = 23$ (bzw. X) ist.

Natürlich kommen im Klartext E, R, N usw. häufiger vor als Q und Y. Aber da jede Zahl von 00 bis 25 als k_4 **gleich wahrscheinlich** ist,

. . . . . .

. . . . . .

Würde der Klartext stattdessen FRUEH lauten, käme beim Schlüssel von oben (EOVKL) natürlich ein anderer Chiffretext heraus. **Aber:** Es gäbe ebenso eine Schlüssel (nämlich ZYFTH), der den Klartext FRUEH zum Chiffretext von oben EPZXO verschlüsseln würde.

Klartext:	FRUEH	=	05	17	20	04	07	
Schlüssel:	ZYFTH	=	25	24	05	19	07	$\oplus_{26}$
Chiffretext:	EPZXO	=	04	15	25	23	14	

Klartext:	WARTE	=	22	00	17	19	04	
Schlüssel:	IPIEK	=	08	15	08	04	10	$\oplus_{26}$
Chiffretext:	EPZXO	=	04	15	25	23	14	

Das hat zur Folge, dass Eve, die den Chiffretext EPZXO abgefangen hat, nicht wissen kann, welcher der (gleich wahrscheinlichen!) Schlüssel (EOVKL, ZYFTH, IPIEK, . . .) zu diesem Chiffretext geführt hat und damit auch nicht, welcher Klartext dahinter steckt (ABEND, FRUEH, WARTE, PASSE, WAFFE, NICHT, DURST, FRODO, . . .).

Diskussion des One-Time-Pads

Unter der Voraussetzung, dass der Schlüssel mit guten Zufallszahlengeneratoren erzeugt wird, ist das One-Time-Pad *perfekt sicher*. Es sind überhaupt keine Angriffe darauf möglich. Selbst ein Chosen-Plaintext-Angriff offenbart mir nur den Teil des Schlüssels, mit dem dieser vorgegebene Klartext verschlüsselt wurde. Dieser wird aber nie mehr verwendet!

Ein Brute-Force-Angriff ist praktisch unmöglich wegen der Zahl der Möglichkeiten, aber auch theoretisch nutzlos, weil es für *jeden* Text dieser Länge einen Schlüssel gibt, der daraus diesen Chiffretext gemacht hätte!

Würde Eve also mittels Brute-Force Schlüssel durchprobieren und auf einen plausiblen Klartext stoßen

. . . . . .

. . . . . .

Laien vermuten ja oft, dass Kryptographie mit komplexer Mathematik zu tun hat. Das One-Time-Pad zeigt, dass sogar *perfekte Sicherheit* mit der einfachsten mathematischen Operation (XOR) erlangt werden kann.[10]

Das Unternehmen *Mils Electronic* war auf die Herstellung von One-Time-Pad-Hardwarelösungen spezialisiert.[11]

Wenn es ein *perfekt sicheres* Verfahren gibt, warum wird es dann nicht verwendet?

- Weiterhin ungelöst ist, wie bei allen symmetrischen Verfahren, der

  . . . . . .

  . . . . . .

  . . . . . .

  . . . . . .

  . . . . . .

- Die Erzeugung guter Zufallszahlen ist schwerer als man vermuten würde.
- Gehen Nachrichten verloren, hat man das Problem der Synchronisation. Der Empfänger muss genau wissen, zu welchem Schlüsselzeichen das Chiffretextzeichen gehört.

Aufgabe 26:
Begründe, wieso bei jedem Zeichen wieder nur Caesar-Verschiebungen verwendet werden und keine allgemeinen monoalphabetischen Substitutionen.

. . . . . .

. . . . . .

[10] Der Beweis der perfekten Sicherheit erfordert natürlich Mathematik, der Entwurf und das Testen von Zufallszahlengeneratoren ebenfalls uvm.

[11] https://www.cryptomuseum.com/manuf/mils/index.htm

Aufgabe 27:
Begründe, wieso die Zeichen des Schlüssel nicht wiederverwendet werden dürfen.

.....

.....

.....
Begründe, wieso die Zeichen des Schlüssel perfekt zufällig und unabhängig sein müssen.

.....

Aufgabe 28:
Lass dir von der Lehrperson die Verschlüsselung mit dem One-Time-Pad im Python-Skript vorführen.

Lass dir dann den Angriff *Crib drag* erklären und vorführen.

Denke dir einen Text geeigneter Länge aus, den du „schlecht“ verschlüsselst und an deine Partnerin/deinen Partner schickst.

Versuche per Crib drag den Text deiner Partnerin/deines Partners zu knacken.