Vigenère brechen - und verbessern - mit einfachen Mitteln

Sean Leichtle GymInf - Fachdidaktik I Herbstsemester 2023

Es handelt sich bei dem vorliegenden Text um eine Unterrichtssequenz im Umfang von ca. 4 Lektionen zur Erweiterung des Lehrmittels «Data Science und Sicherheit», Kapitel 2 «Geheimschriften und Datensicherheit». Dieses Material wurde im Herbstsemester 2023 als Leistungsnachweis für das Modul «Fachdidaktik Informatik I» an der ETH Zürich im Rahmen des GymInf-Programms erarbeitet.

Die Unterrichtssequenz thematisiert den Angriff auf die Vigenère-Chiffre mittels der Autokorrelation zur Bestimmung der Schlüssellänge. Sie ist deshalb relevant, weil die Kryptanalyse anhand der Autokorrelation eine Alternative zu den Friedman- sowie Kasiski-Tests bietet, die in dem erwähnten Lehrmittel nicht behandelt wird. Ferner ist die Autokorrelation einfach zu implementieren und kann auch dann die Schlüssellänge ermitteln, wenn es keine sich wiederholenden N-Gramme im Geheimtext gibt.

Im ersten Teil dieser Unterrichtssequenz wird die Bestimmung der Schlüssellänge behandelt, im zweiten Teil deren algorithmische Umsetzung. Bei dieser Umsetzung handelt es sich um vier sehr einfache Funktionen, an die die SuS unter Beachtung des Modularitätsprinzips herangeführt werden. Im abschliessenden Teil wird eine Verbesserung der Chiffre behandelt (die Autokey-Chiffre), die für den Angriff durch die Autokorrelation nicht anfällig ist. Es befinden sich im Anhang zwei Funktionen, die für die Lehrperson bestimmt sind und die eine graphische Darstellung der Autokorrelation ermöglichen.[1]

Voraussetzungen

Es wird von den folgenden Voraussetzungen ausgegangen:

- Die SuS können Texte anhand der Vigenère-Chiffre chiffrieren und dechiffrieren.
- Die SuS können bei bekannter Schlüssellänge mittels der Häufigkeitsanalyse einen nach dem Vigenère-Verfahren chiffrierten Text entschlüsseln.
- Die SuS können mittels des Kasiski-Tests Vorschläge zur Schlüssellänge erarbeiten.
- Die SuS können for-Schleifen zusammen mit der range( )-Funktion verwenden.
- Die SuS können den Befehl «len( )» einsetzen, um die Länge eines Strings zu bestimmen.

Lernziele

Die Unterrichtssequenz hat den folgenden Bezug zum Rahmenlehrplan:

- «Verschiedene Codierungen und Darstellungen von Informationen kennen
- Sicherheitsaspekte der digitalen Kommunikation verstehen, z.B. Verschlüsselung [...]
- Eigene und fremde Lösungswege formal beschreiben und kritisch analysieren
- Algorithmen entwerfen, beurteilen und in einer Programmiersprache umsetzen»

(EDK, «Rahmenlehrplan für die Maturitätsschulen: Informatik», 2017, S. 4)

Ferner sind für diese Sequenz die folgenden Lernziele anvisiert:

- Die SuS verstehen die Grenzen des Kasiski-Tests zur Bestimmung der Schlüssellänge.
- Die SuS verstehen die Autokorrelation als kryptanalytische Methode und können sie anwenden, um die Länge des Schlüssels zu bestimmen, anhand dessen ein Geheimtext nach Vigenère chiffriert wurde.
- Die SuS können einen Geheimtext ohne Angabe der Schlüssellänge vollständig entschlüsseln.
- Die SuS können ein algorithmisches Verfahren für die Autokorrelation entwerfen und dieses in Python implementieren.
- Die SuS können Texte anhand der Autokey-Chiffre chiffrieren und dechiffrieren.
- Die SuS verstehen die Autokey-Chiffre als eine Weiterentwicklung der Vigenère-Chiffre, die den Angriff mittels Autokorrelation unterbindet.
- Die SuS können die Schwäche der Autokey-Chiffre erkennen und benennen.

[1] Alle Funktionen stehen auch im folgenden GitHub-Repository zur Verfügung: https://github.com/sean-leichtle/Autocorrelation.

Hinweis zu den verwendeten Schriftarten

Im Folgenden erscheinen Lösungen zu den Aufgaben auf einem grauen Hintergrund und werden in Rot dargestellt (z.B. Schlüssellänge: 3). Funktionsnamen sowie Code-Elemente verwenden die folgende Schriftart: `text_verschieben(text, stellen)`. Komplette Funktionen werden folgendermassen dargestellt:

```
1 def texte_vergleichen(text1, text2):
```

0. Einstieg

Der folgende Text wurde nach dem Vigenère-Verfahren verschlüsselt:

BAMQEGSEKVIMVNTLFKZCAKIZWRXLDXWUXIDBVWXCT

Aufgaben:

1. Versuche anhand des Kasiski-Tests die Länge des verwendeten Schlüssels zu ermitteln. Was fällt hier auf?

2. Wie könnte man die Schlüssellänge andernfalls bestimmen?

1. Es gibt keine sich wiederholenden Buchstabensequenzen.
2. <Individuelle Lösungen>

1. Die Bestimmung der Schlüssellänge durch die Autokorrelation

1.1. Autokorrelation: Das Prinzip

Die Lösung des Problems ist denkbar einfach:

- Man nimmt eine Kopie des Geheimtexts, legt sie unter den ursprünglichen Text um eine Stelle nach rechts verschoben und notiert die Anzahl an übereinstimmenden Buchstaben.
- Dann wird die Kopie um noch eine Stelle verschoben und die übereinstimmenden Buchstaben werden wieder notiert.
- Dieser Prozess wird dann je nach Länge des Texts 30 bis 50 Male wiederholt.

Es ist dabei zu beachten, dass es keine nennenswerte Anzahl an Übereinstimmungen geben wird, falls die Kopie um fast so viele Stellen verschoben wird, wie der Geheimtext Buchstaben hat.

Das folgende Bild illustriert das Prinzip für die Klartextnachricht «GUTENMORGEN»:

Verschiebungen

```
 0     :  GUTENMORGEN
 1     :  -GUTENMORGE
 2     :  --GUTENMORG
 3     :  ---GUTENMOR
 4     :  ----GUTENMO
 5     :  -----GUTENM
 6     :  ------GUTEN
 7     :  -------GUTE
 8     :  --------GUT
 9     :  ---------GU
10     :  ----------G
```

Aufgaben:

3. Wende die Autokorrelation an dem folgenden Geheimtext an. Schreibe zuerst den Text auf zwei verschiedene Blätter. Lege dann eine Kopie unter die andere. Dann verschiebe den unteren Text sukzessiv um eine Stelle nach rechts und notiere für jede Verschiebung jeweils die Zahl an übereinstimmenden Buchstaben:

   BAMQEGSEKVIMVNTLFKZCAKIZWRXLDXWUXIDBVWXCT

4. Skizziere ein Diagramm, bei dem die Anzahl an Verschiebungen auf der X-Achse und die Anzahl an Übereinstimmungen auf der Y-Achse dargestellt werden. Vergleiche die Anzahl an Übereinstimmungen pro Verschiebung. Was fällt hier auf? Können wir anhand dieser Information Rückschlüsse auf die Schlüssellänge ziehen?

5. Erkläre warum man mit dieser Information in der Lage ist, die Schlüssellänge abzuschätzen.

6. Versuche anhand der Ergebnisse von Aufgabe 4 den Schlüssel zu ermitteln und den Geheimtext durch eine Häufigkeitsanalyse von Buchstabengruppen zu dechiffrieren.

3.

0: 41	5: 1	10: 0	15: 0	20: 1	25: 0	30: 0	35: 1	40: 0
1: 0	6: 4	11: 1	16: 0	21: 0	26: 1	31: 0	36: 0	
2: 0	7: 1	12: 3	17: 0	22: 0	27: 1	32: 0	37: 0	
3: 4	8: 0	13: 2	18: 0	23: 1	28: 0	33: 0	38: 0	
4: 1	9: 3	14: 0	19: 1	24: 1	29: 0	34: 0	39: 0	

4. Es zeichnen sich Maxima bei den Verschiebungen 3, 6, 9 und 12 ab, die auf die Schlüssellänge 3 hindeuten.

5. Im Klartext kommen bestimmte Zeichen mehrfach vor. Es ist anzunehmen, dass diese Zeichen manchmal durch die gleichen Schlüsselbuchstaben chiffriert werden, sodass die Zahl an Übereinstimmungen dort am grössten ist, wo die Kopie des Chiffretexts um ein Vielfaches er Schlüssellänge verschoben worden ist.

6. Nach der Aufteilung des Geheimtexts in Sequenzen der Länge 3 werden die jeweiligen Buchstabengruppen gebildet und die Schlüsselteile durch eine Häufigkeitsanalyse ermittelt.

   BAM QEG SEK VIM VNT LFK ZCA KIZ WRX LDX WUX IDB VWX CT

   Gruppe 1: B Q S V V L Z K W L W I V C
   Gruppe 2: A E E I N F C I R D U D W T
   Gruppe 3: M G K M T K A Z X X X B X

   Häufigkeiten Gruppe 1: V: 3 → R, L: 2 → H, W: 2 → S
   Häufigkeiten Gruppe 2: D: 2 → Z, E: 2 → A, I: 2 → E
   Häufigkeiten Gruppe 3: X: 4 → T, K: 2 → G, M: 2 → I

   Schlüssellänge: 3
   Schlüssel: RAT

   Klartext: KATZENBEREITENAUFRICHTIGFREUDEFUERDIEWELT

1.2. Autokorrelation: Die Theorie

Der Begriff «Autokorrelation» entstammt der Stochastik bzw. der Signalverarbeitung und bezeichnet die Korrelation eines Signals mit einer (zeitlich) verschobenen Kopie von sich selbst (Wikipedia, 2023). Die Korrelationsfunktion $C(t)$ misst dann die Ähnlichkeit zwischen einer Sequenz s und einer Verschiebung von

s um t Positionen (Menezes et al., S. 180). Es gibt unterschiedliche - teils aufwendige - Ansätze, diese Ähnlichkeit zu messen. Wir können allerdings auf diese Ansätze verzichten, denn wir müssen für die Kryptanalyse von Vigenère nur die Anzahl an übereinstimmenden Buchstaben pro Verschiebung messen.

Warum funktioniert die Autokorrelation? Im Einzelnen wird wie beim Kasiski-Test angenommen, dass bestimmte Buchstaben im Geheimtext immer wieder durch denselben Buchstaben des Schlüssels chiffriert worden sind. Anders als beim Kasiski-Test sucht man aber nicht nach sich wiederholenden Sequenzen wie Bi- oder Tri-Grammen. Stattdessen wird die Tatsache ausgenutzt, dass sich Maxima bei Vielfachen der Schlüssellänge abzeichnen, denn genau an diesen Stellen wird der Klartext mit demselben Buchstaben des Schlüssels chiffriert. Es bleibt nur noch, die Anzahl an übereinstimmenden Zahlen pro Verschiebung zu zählen und die Maxima zu identifizieren.

Im Kontext der Kryptanalyse hat die Autokorrelation einige Vorteile. Im Unterschied zum Friedman- bzw. Kasiski-Test ist die Kryptanalyse durch Autokorrelation effizienter und anschaulicher (vgl. CrypTool-Online). Sie ist vergleichsweise einfach zu implementieren und zudem nicht auf sich wiederholende Sequenzen (wie Bi- oder Tri-Gramme) im Geheimtext angewiesen. Wie wir schon bei Aufgabe 1 gesehen haben, sind Geheimtexte ohne solche Sequenzen durchaus möglich, wenn auch bei zunehmender Länge der zu übermittelnden Botschaft unwahrscheinlicher.

2. Algorithmische Umsetzung

2.1. Erste Vorüberlegung

Das Wort «Verschiebungen» in diesem Kontext erweckt die Vorstellung, dass wir bei der algorithmischen Umsetzung der Autokorrelation tatsächlich eine Kopie eines Geheimtexts unterhalb des Geheimtexts selbst in Bewegung setzen müssen. Die «händische» Dechiffrierung des Geheimtexts in den Aufgaben 3, 4 und 6 trägt auch zu dieser Vorstellung bei.

Man kann aber einfacher vorgehen! Wir versuchen dies jetzt, ohne dabei auf irgendwelche Python-Befehle zu achten.

Aufgabe:

7. Versuche, anhand des oben stehenden Ergebnisses für den Text «GUTENMORGEN» genau zu beschreiben, wie die Ausgabe entstanden ist.

7.

- Für jede Zeile wird eine Anzahl Geviertstriche, die der Zahl an Verschiebungen entspricht, dem Text vorangestellt. Eine entsprechende Anzahl an Buchstaben wird am Ende des Texts abgeschnitten.
- Die so umgestalteten Texte werden dann nacheinander ausgegeben, und zwar geordnet nach der zunehmenden Anzahl an Verschiebungen.

2.2. Erste Umsetzung

Bei der Implementierung in Python können wir die Beschreibungen / Beobachtungen von Aufgabe 7 verwenden. Einzig die Frage nach der passenden Syntax ist nun noch zu überlegen.

Wir wissen aus dem bisherigen Unterricht, dass man einen String in Python mit «*» vervielfältigen kann. Will man in einer Funktion «333» zurückgeben, so kann man den folgenden Befehl verwenden:

```
...
return "3" * 3
```

Man kann auch Teile von bestehenden Strings zurückgeben lassen.

- Analog zu Listen kann man einen String «s» anhand seiner Indexe mit der Syntax «`[A:E]`» ansprechen.

- `A` bezeichnet hier den ersten und `E` den letzten Index, wobei sich der range von `A` bis zur Stelle vor `E` erstreckt.
- Um die ersten drei Buchstaben eines Strings zu drücken, verwendet man also den folgenden Befehl:

```
...
s = "HALLO"
return s[0:3]   # Ergebnis: HAL
```

- Letztlich soll man nicht vergessen, dass man durch «+» Strings zusammenfügen kann.

Bezüglich der Ausgabe von sukzessiv verschobenen Texten gäbe es folgende Möglichkeiten:

- Man könnte zunächst eine Funktion schreiben, die einen Text eine übergebene Anzahl an Stellen verschiebt.
- Diese Funktion könnte als Unterprogramm in einer anderen Funktion verwendet werden, die anhand einer for-Schleife die Anzahl an Verschiebungen sukzessiv erhöht.

Aufgaben:

8. Schreibe eine Funktion «`text_verschieben(text, stellen)`», die einen *Text* und eine Anzahl an *Stellen* entgegennimmt. Die Funktion soll dann den *Text* zurückgeben, und zwar um eine Anzahl an Stellen verschoben und mit einer Anzahl an Geviertstriche vorangestellt, die jeweils der übergebenen Zahl *Stellen* entspricht.

9. Schreibe eine Funktion «`verschiebungen_ausgeben(text)`», die einen *Text* entgegennimmt und alle Verschiebungen des *Texts* von 0 Stellen bis zu einer Anzahl an Stellen ausgibt, die der Textlänge entspricht. Verwende dabei das Ergebnis von Aufgabe 8 als Unterprogramm.

10. Teste die Funktion «`verschiebungen_ausgeben(text)`» mit dem Geheimtext «BAMQEGSEKVIMVNTLFKZCAKIZWRXLDXWUXIDBVWXCT» als Eingabe. Das Ergebnis sollte dem folgenden Bild in etwa entsprechen:

Verschiebungen

```
0       :  BAMQEGSEKVIMVNTLFKZCAKIZWRXLDXWUXIDBVWXCT
1       :  -BAMQEGSEKVIMVNTLFKZCAKIZWRXLDXWUXIDBVWXC
2       :  --BAMQEGSEKVIMVNTLFKZCAKIZWRXLDXWUXIDBVWX
3       :  ---BAMQEGSEKVIMVNTLFKZCAKIZWRXLDXWUXIDBVW
4       :  ----BAMQEGSEKVIMVNTLFKZCAKIZWRXLDXWUXIDBV
5       :  -----BAMQEGSEKVIMVNTLFKZCAKIZWRXLDXWUXIDB
6       :  ------BAMQEGSEKVIMVNTLFKZCAKIZWRXLDXWUXID
7       :  -------BAMQEGSEKVIMVNTLFKZCAKIZWRXLDXWUXI
8       :  --------BAMQEGSEKVIMVNTLFKZCAKIZWRXLDXWUX
9       :  ---------BAMQEGSEKVIMVNTLFKZCAKIZWRXLDXWU
10      :  ----------BAMQEGSEKVIMVNTLFKZCAKIZWRXLDXW
11      :  -----------BAMQEGSEKVIMVNTLFKZCAKIZWRXLDX
12      :  ------------BAMQEGSEKVIMVNTLFKZCAKIZWRXLD
13      :  -------------BAMQEGSEKVIMVNTLFKZCAKIZWRXL
14      :  --------------BAMQEGSEKVIMVNTLFKZCAKIZWRX
15      :  ---------------BAMQEGSEKVIMVNTLFKZCAKIZWR
16      :  ----------------BAMQEGSEKVIMVNTLFKZCAKIZW
17      :  -----------------BAMQEGSEKVIMVNTLFKZCAKIZ
18      :  ------------------BAMQEGSEKVIMVNTLFKZCAKI
19      :  -------------------BAMQEGSEKVIMVNTLFKZCAK
20      :  --------------------BAMQEGSEKVIMVNTLFKZCA
21      :  ---------------------BAMQEGSEKVIMVNTLFKZC
22      :  ----------------------BAMQEGSEKVIMVNTLFKZ
23      :  -----------------------BAMQEGSEKVIMVNTLFK
24      :  ------------------------BAMQEGSEKVIMVNTLF
25      :  -------------------------BAMQEGSEKVIMVNTL
26      :  --------------------------BAMQEGSEKVIMVNT
27      :  ---------------------------BAMQEGSEKVIMVN
28      :  ----------------------------BAMQEGSEKVIMV
29      :  -----------------------------BAMQEGSEKVIM
30      :  ------------------------------BAMQEGSEKVI
31      :  -------------------------------BAMQEGSEKV
32      :  --------------------------------BAMQEGSEK
33      :  ---------------------------------BAMQEGSE
34      :  ----------------------------------BAMQEGS
35      :  -----------------------------------BAMQEG
36      :  ------------------------------------BAMQE
37      :  -------------------------------------BAMQ
38      :  --------------------------------------BAM
39      :  ---------------------------------------BA
40      :  ----------------------------------------B
```

8.

```
def text_verschieben(text, stellen):
    """
    Gibt den Eingabetext um eine Anzahl
    an Stellen "nach rechts" verschoben
    zurück
    """
    return (stellen * "-") + text[0:len(text) - stellen]
```

9.

```
def verschiebungen_ausgeben(text):
    """
    Eine Hilfsfunktion, die über die Länge
    des Eingabetexts iteriert und alle
    Textverschiebungen ausgibt
    """
    print("Verschiebungen")
    print("--------------")
    for i in range(len(text)):

        st = text_verschieben(text, i)

        if(i < 10):
            print("\t ", i, "\t: ", st)
        else:
            print("\t", i, "\t: ", st)
```

(Auf die if-else-Anweisung in den Zeilen 13-16 der zweiten Funktion kann verzichtet werden, denn sie dient lediglich der «Verschönerung» der Ausgabe. In dem Fall sollte man stattdessen lediglich den print-Befehl in Zeile 16 verwenden.)

10. <Die Ausgaben individueller Lösungen werden überprüft.>

2.3. Zweite Vorüberlegung

Bei Aufgabe 3 musste man pro Verschiebung die Anzahl an übereinstimmenden Buchstaben notieren. Das Ergebnis sah dann ungefähr so aus:

0: 41	5: 1	10: 0	15: 0	20: 1	25: 0	30: 0	35: 1	40: 0
1: 0	6: 4	11: 1	16: 0	21: 0	26: 1	31: 0	36: 0	
2: 0	7: 1	12: 3	17: 0	22: 0	27: 1	32: 0	37: 0	
3: 4	8: 0	13: 2	18: 0	23: 1	28: 0	33: 0	38: 0	
4: 1	9: 3	14: 0	19: 1	24: 1	29: 0	34: 0	39: 0	

(Hier sollte man die Angabe 41 für 0 Verschiebungen ignorieren, sie gibt nur die Länge des Texts wieder.)

Die Frage ist nun, wie die Berechnung dieses Ergebnisses implementiert werden kann.

Aufgabe:

11. Versuche analog zu Aufgabe 7, ohne auf die Python-Syntax zu achten, genau zu beschreiben, wie die eben dargestellte Ausgabe zustande gekommen ist.

11.

- Man vergleicht die gleichen Indexe sowohl eines Ausgangstexts als auch die eines um eine Stelle verschobenen Texts.
- Man notiert die Anzahl an Übereinstimmungen.
- Dann wiederholt man den Vergleich zwischen einem Ausgangstext und einem um zwei Stellen verschobenen Text usw.

2.4. Zweite Umsetzung

Bei der Implementierung in Python müssen wir lediglich die Ergebnisse von Aufgabe 11 syntaktisch umsetzen. Dabei wollen wir uns ein zentrales Prinzip der Programmierung zu eigen machen: die Modularität. Modular zu programmieren, bedeutet, dass man komplexere Programme aus kleineren,

einfacheren Programmen oder «Modulen» komponiert (vgl. Arnold et al. 24). Die text_verschieben-Funktion von Aufgabe 8 ist ein Paradebeispiel eines solchen sehr einfachen Moduls.

Warum sollte man so vorgehen? Mindestens zwei sehr grosse Vorteile liegen auf der Hand. Auf der einen Seite dient die Modularität der Wiederverwendbarkeit. Es ist natürlich möglich, ein sehr komplexes Programm zu schreiben, ohne auf die Modularität zu achten. Aber dann sind die Funktionalitäten, die das Programm beinhaltet, an das Programm gebunden und können nicht unabhängig von diesem Programm verwendet werden. Viel besser wäre es, diese Funktionalitäten von vornherein in primitiveren Modulen zur Verfügung zu stellen, denn dann können wir sie für viele Zwecke einsetzen.

Auf der anderen Seite zwingt uns der modulare Programmentwurf dazu, komplexe Aufgaben in kleinere und - idealerweise - kleinste logische Teilaufgaben herunterzubrechen. Dadurch versetzen wir uns in die Lage, sehr schwierige Problemstellungen systematisch angehen zu können.

Aufgaben:

12. Schreibe eine Funktion «`texte_vergleichen(text1, text2)`», die zwei *Texte* der gleichen Länge entgegennimmt. Die Funktion soll die Index-Stellen 0 … n vergleichen und die Zahl an übereinstimmenden Buchstaben zurückgeben.

13. Schreibe eine Funktion «`autokorrelation(text, verschiebungen)`», die einen *Text* sowie eine *Obergrenze an Verschiebungen* entgegennimmt und die Anzahl an Übereinstimmungen pro Verschiebung ausgibt. Dabei soll diese Funktion im Sinne der Modularität die text_verschieben- sowie texte_vergleichen-Funktionen als Unterprogramme verwenden.

14. Teste die Funktion «`autokorrelation(text, verschiebungen)`» mit dem Geheimtext «BAMQEGSEKVIMVNTLFKZCAKIZWRXLDXWUXIDBVWXCT» als Eingabe. Das Ergebnis sollte mit dem Ergebnis von Aufgabe 3 übereinstimmen.

12.

```python
def texte_vergleichen(text1, text2):
    """
    Gibt die Anzahl übereinstimmender
    Buchstaben zwischen text1 und text2
    zurück
    """
    count = 0

    for i in range(len(text1)):
        if(text1[i] == text2[i]):
            count += 1

    return count
```

13.

```python
def autokorrelation(text, verschiebungen):
    """
    Gibt die Anzahl übereinstimmender
    Buchstaben je Verschiebung vom
    Eingabetext bis zu einer übergebenen
    Höchstzahl an Verschiebungen
    zurück
    """
    for i in range(verschiebungen):

        vt = text_verschieben(text, i)

        count = texte_vergleichen(text, vt)

        print(i, ": ", count)
```

14. <Die Ausgaben individueller Lösungen werden überprüft.>

2.5. Anwendung / Wissenssicherung

Gegeben sei der mit dem Vigenère-Verfahren chiffrierte Text:

JICHKWEJKJPMZIVBZICUXEPHZITOKRCPZIYIAXTNSICXKRYFXECCKMEFZ

Aufgaben:

15. Bestimme mit der autokorrelation-Funktion die Länge des Schlüssels.

16. Entschlüssele den Geheimtext mithilfe der Schlüssellänge und der Häufigkeitsanalyse.

15. Die Schlüssellänge ist 4.

16. Nach Aufteilung des Geheimtexts in Sequenzen der Länge 4 werden die jeweiligen Buchstabengruppen gebildet und die Schlüsselteile durch die Häufigkeitsanalyse wie folgt ermittelt:

JICH KWEJ KJPM ZIVB ZICU XEPH ZITO KRCP ZIYI AXTN SICX KRYF XECC KMEF Z

Gruppe 1: J K K Z Z X Z K Z A S K X K Z
Gruppe 2: I W J I I E I R I X I R E M
Gruppe 3: C E P V C P T C Y T C Y C E
Gruppe 4: H J M B U H O P I N X F C F

Häufigkeiten Gruppe 1: K: 5 → G, Z: 5 → V
Häufigkeiten Gruppe 2: I: 6 → E
Häufigkeiten Gruppe 3: C: 5 → Y, E: 2 → A, P: 2 → L, T: 2 → P, Y: 2 → U
Häufigkeiten Gruppe 4: F: 2 → B, H: 2 → D

Schlüssel: GELB

Klartext: DERGESTIEFELTEKATERTRAEGTEINENROTENHUTIMMERWENNERARBEITET

2.6. Transfer

Die Vigenère-Chiffre ist offensichtlich mit einfachen Mitteln zu brechen. Es gilt jetzt zu überlegen, ob sie verbessert werden kann, und wenn ja, wie diese Verbesserung aussehen könnte. Diese Frage beantworten zu können, setzt allerdings eine Sensibilisierung für den Aspekt der Chiffre voraus, der von der Autokorrelation ausgenutzt wird.

Aufgaben:

17. Was ist die Schwachstelle der Vigenère-Chiffre, die von der Autokorrelation offengelegt wird?

18. Wie könnte man diese Schwachstelle schliessen?

17. Der Schlüssel wiederholt sich periodisch.
18. Der Schlüssel ist so zu wählen, dass er sich nicht periodisch wiederholt.

3. Die Autokey-Chiffre

Praktischerweise hat Blaise de Vigenère ein Verfahren dieser Art erfunden. Die Autokey-Chiffre markiert eine kryptographische Neuerung, die den Klartext selbst einsetzt, um den Schlüsselstrom zu erzeugen und dadurch jegliche Periodizität im Geheimtext zu unterbinden (vgl. Bauer, 159). Somit ist die Chiffre ein kontextsensitives Verfahren, denn der Verschlüsselungsvorgang (und somit der Chiffretext) wird massgeblich vom Klartext selbst bestimmt.

Wie die Vigenère-Chiffre verwendet die Autokey-Chiffre das Vigenère-Quadrat. Die Chiffre verwendet auch ein Wort, den sogenannten «Primer», das dem Schlüsselwort der Vigenère-Chiffre entspricht. Der Primer ist allerdings kein Schlüsselwort, vielmehr bildet der Primer den Anfang des eigentlichen Schlüssels und wird nur einmal unter die Klartextnachricht geschrieben. Dort, wo man bei der Vigenère-Chiffre wiederholt das Schlüsselwort verwendet hätte, schreibt man bei der Autokey-Chiffre nach dem Primer die Buchstaben der Klartextnachricht selbst. Im Anschluss wird dann anhand des Vigenère-Quadrats wie gewohnt verschlüsselt.

Es folgt ein Beispiel mit dem Primer «SCHWARZ»:

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

Der Dechiffrierungsvorgang funktioniert wie bei der Vigenère-Chiffre umgekehrt. Man schreibt zuerst den Primer unter den Anfang des Chiffriertexts. Anhand des Vigenère-Quadrats entschlüsselt man den ersten Teil des Chiffretexts. Dabei fügt man jeden dechiffrierten Buchstaben sukzessiv dem Primer zu und setzt so den Dechiffrierungsvorgang fort.

Das folgende Beispiel illustriert diesen Prozess in Auszügen:

Geheimtext:	C C M B E V T W Z J M R Y B D V E L U L L E Z D I Y G E H V H
Schlüssel:	S C H W A R Z K
Geheimtext:	K

Geheimtext:	C C M B E V T W Z J M R Y B D V E L U L L E Z D I Y G E H V H
Schlüssel:	S C H W A R Z K A
Geheimtext:	K A

Geheimtext:	C C M B E V T W Z J M R Y B D V E L U L L E Z D I Y G E H V H
Schlüssel:	S C H W A R Z K A F
Geheimtext:	K A F

…

Geheimtext:	C C M B E V T W Z J M R Y B D V E L U L L E Z D I Y G E H V H
Schlüssel:	S C H W A R Z K A F F E E U M Z
Geheimtext:	K A F F E E U M Z

…

Aufgaben:

19. Wir bilden Zweiergruppen. Wählt ein Schlüsselwort und chiffriert eine kurze Nachricht mit der Autokey-Chiffre.

20. Tauscht eure Nachrichten und Schlüssel mit einer anderen Gruppe aus und entschlüsselt diese Nachrichten.

19. <Individuelle Lösungen>
20. <Individuelle Lösungen>

Es bleibt zu überprüfen, ob die Autokey-Chiffre tatsächlich imstande ist, einen Angriff durch die Autokorrelation abzuwehren. Zu diesem Zweck gehen wir von der folgenden Klartextnachricht sowie von dem folgenden Schlüssel aus:

Klartext: KAFFEEUMZEHNUHRWAEHRENDDERPAUSE
DANNBESPRECHENWIRDASWEITEREVORGEHEN

Schlüssel: GRUEN

Aufgaben:

21. Verschlüsselt diese Nachricht nach dem Vigenère- und dem Autokey-Verfahren.

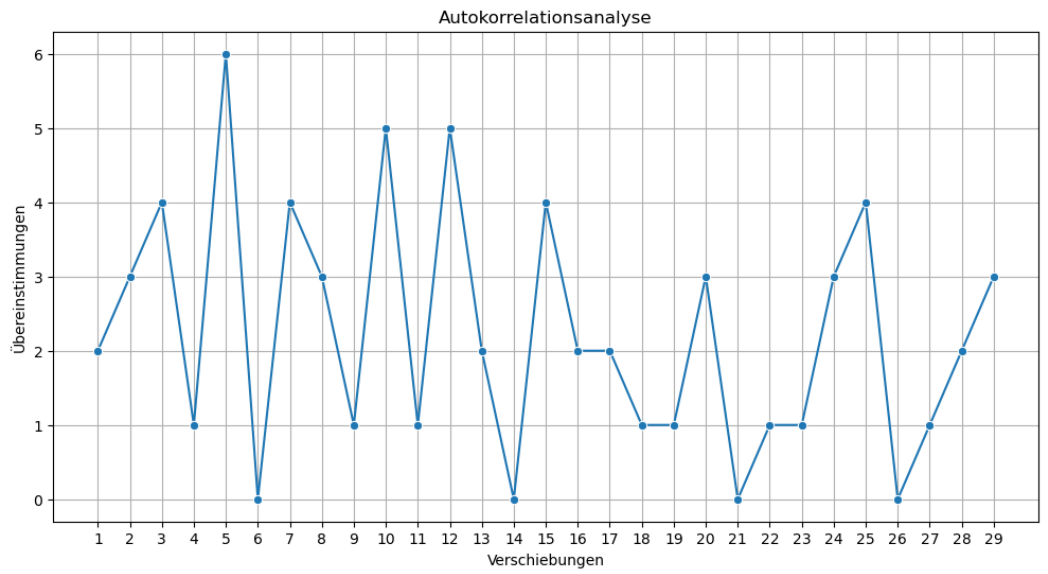
22. Versuche mit der autokorrelation-Funktion die Länge des Schlüssels bzw. des Primers für den jeweiligen Chiffretext zu bestimmen.

23. Welche möglichen Schwachstellen gibt es bei der Autokey-Chiffre?

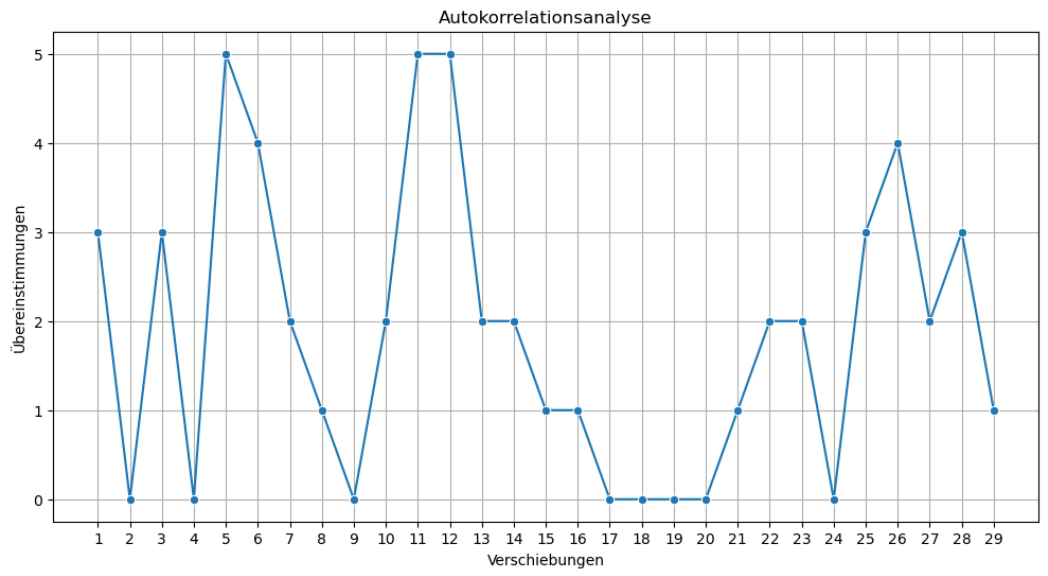
21. Vigenère: QRZJRKLGDRNEOLECRYLEKEXHRXGUYFKUU
RAHVMTEKTBIACZLHNYNYMGKIYZBXXYLRT

Autokey: QRZJROUREILHGGVDNYOIANHKVVCDXWVSA
HFFHSCEFGZTEAKYHNOEVLTWNIDHVXICSE

22. Ergebnis der Anwendung der autokorrelation-Funktion auf die Vigenère-verschlüsselte Nachricht:



Ergebnis der Anwendung der autokorrelation-Funktion auf die Autokey-verschlüsselte Nachricht:



23: Der Schlüssel besteht zum Teil aus dem Klartext. Wenn die Sprache des Klartexts bekannt ist, ist es also möglich, verschiedene häufig vorkommende Wörter dieser Sprache an verschiedenen Stellen des Schlüssels einzusetzen und entsprechend zu dechiffrieren, bis ein Klartext erscheint.

Fazit

Übliche Methoden zur Ermittlung der Schlüssellänge bei Vigenère-chiffrierten Nachrichten hängen von sich wiederholenden Sequenzen im Chiffretext ab (Kasiski-Test), die nicht immer vorhanden sind, oder nehmen relativ komplexe Berechnungen in Kauf (Friedman-Test). Im Unterschied dazu bietet die Autokorrelation eine intuitive Methode zur Bestimmung der Schlüssellänge, die leicht zu implementieren und nicht auf sich wiederholende Sequenzen im Chiffretext angewiesen ist. Durch die wiederholte Verschiebung einer Kopie des Chiffretexts relativ zum Chiffretext ergibt sich jeweils eine Anzahl an übereinstimmenden Buchstaben. Wenn man diese Ergebnisse zusammenträgt, bilden sich klar erkennbare Maxima, die auf die gesuchte Schlüssellänge hindeuten. Da die Autokorrelation die Periodizität des Schlüssels ausnutzt, liegt es nahe, diese Schwachstelle dadurch zu schliessen, dass man bei der Verschlüsselung kein sich wiederholendes Schlüsselwort einsetzt. Ein Verfahren dieser Art ist von Blaise de Vigenère tatsächlich erfunden worden und wird als Autokey-Chiffre bezeichnet. Anstelle eines Schlüsselworts werden ein Primer sowie Teile des Klartexts selbst als Schlüssel eingesetzt, wodurch sich die Chiffre als kontextsensitiv erweist. Die Analyse einer entsprechend chiffrierten Nachricht durch die Autokorrelation ist somit zwecklos.

Anhang

Die folgenden zwei Funktionen erlauben eine graphische Darstellung von Analysen mittels der Autokorrelation. Es handelt sich einerseits um eine leicht abgewandelte Version der autokorrelation-Funktion, die die Übereinstimmungen pro Verschiebung in einer Dictionary/Map-Datenstruktur speichert, andererseits um eine Funktion, die ein Liniendiagramm zeichnet.

Es wäre denkbar, auch diese Funktionen im Laufe der Unterrichtssequenz implementieren zu lassen. Dies würde allerdings voraussetzen, dass die SuS mit Dictionaries vertraut sind. Ebenso würde eine Auseinandersetzung mit den notwendigen Python-Bibliotheken möglicherweise von den gesteckten Unterrichtszielen ablenken.

```
def autokorrelation2(text, verschiebungen):
    """
    Zählt übereinstimmende Buchstaben
    zwischen Eingabe- und verschobenen
    Texte bis zu einer übergebenen
    Zahl an Verschiebungen und gibt
    das Ergebnis als Dictionary
    zurück
    """
    uebereinstimmungen = {}

    for i in range(verschiebungen):

        vt = text_verschieben(text, i)

        count = texte_vergleichen(text, vt)

        uebereinstimmungen[i] = count

    return uebereinstimmungen
```

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

def autokorrelation_zeigen(dictionary):
    """
    Eine Hilfsfunktion, um
    Übereinstimmungen je Verschiebung
    als Line-Chart darzustellen
    """

    # entfernt erstes key-value-Paar, um
    # Verzehrung des Graphs zu vermeiden
    dictionary.pop(0)

    plt.figure(figsize=(12, 6))
    df = pd.Series(dictionary)
    new_sns = sns.lineplot(data = df, marker="o")
    new_sns.set(xticks=df.keys())

    plt.title('Autokorrelationsanalyse')
    plt.xlabel('Verschiebungen')
    plt.ylabel('Übereinstimmungen')

    plt.grid()
    plt.show()
```

Die zwei vorangehenden Funktionen können durch den folgenden Aufruf verwendet werden, wobei der zu analysierende Text dem sample-Variable zugewiesen werden muss. Auch die Anzahl an übergebene Verschiebungen muss ggf. geändert werden.

```
sample = ""

autokorrelation_zeigen(autokorrelation2((sample), 30))
```

Literaturverzeichnis

Arnold, J., Donner, C., Hauser, U., Hauswirth, M., Hromkovič, J., Kohn, T, Komm, D., Maletinsky, D., Roth, N. (2021/2023): *INFORMATIK. Programmieren und Robotik*. Baar: Klett und Balmer.

Barot, M., Dorn, B., Fourny, G., Gallenbacher, J., Hromkovič, J., Lacher, R. (2022): *INFORMATIK. Data Science und Sicherheit*. Baar: Klett und Balmer.

Bauer, F. (2007[4]): Decrypted Secrets: Methods and Maxims of Cryptology. Berlin und Heidelberg: Springer Verlag.

CrypTool-Online (2023): «Autokorrelation [Reiter «Hintergrund»]». Abgerufen am: 04.11.2023: https://www.cryptool.org/de/cto/autocorrelation.

Dietrich, L. Lautebach, U., Schaller, T. (2017): «Kryptografie – Informatik des Vertrauens (Hintergrundwissen)». Abgerufen am: 07.11.2023: https://lehrerfortbildung-bw.de/u_matnatech/informatik/gym/bp2016/fb1/3_rechner_netze/1_hintergrund/9_krypto/07_run_hintergrund_kryptografie.pdf.

EDK (2017): *Rahmenlehrplan für die Maturitätsschulen: Informatik*. Abgerufen am: 30.10.2023: https://edudoc.ch/record/131917/files/rlp_inf_2017_d.pdf.

Lyons, J. (2009/2012): «Autokey Cipher.» Practical Cryptography. Abgerufen am 16.11.2023: http://www.practicalcryptography.com/ciphers/autokey-cipher/.

Menezes, A., van Oorschot, P., Vanstone, S. (2001[5]): *Handbook of Applied Cryptography*. Boca Raton, Florida: CRC Press.

Wikipedia (2023): «Autokey cipher.» Abgerufen am: 16.11.2023: https://en.wikipedia.org/wiki/Autokey_cipher.