Inhaltsverzeichnis

1. Verschlüsseln

1.1. Ziele

☑ Wir wollen folgende Themen erarbeiten:

- einfache Verschlüsselungsverfahren: Skytale, Cäsar und Vigenère
-
- C,java,C++, php
- Filehandling
- Stringmanipulationen
- 1 und 2 dimensionale Arrays

☑ http://www.inf.fh-flensburg.de/lang/krypto/klassisch.htm

So, genug der Vorbereitung.
HAQ AHA JHRAFPUR VPU IVRY FCNFF VZ HAGREEVPUG (frei nach Cäsar)

1.2. Cäsar-Verfahren

☑ Cäsar-verschlüsseln:
Beim Cäsar-Verfahren wird das gesamte Alphabet um eine bestimmte Anzahl von Buchstaben verschoben und dann
jeder Buchstabe des Klartextes einzeln verändert.
Der "**Key**" besteht also aus einem **Verschiebungsbuchstaben** oder einer

Verschiebungszahl.

Als Beispiel nehmen wir den Klartext **Diese Nachricht ist geheim!** und den Verschiebebuchstaben "**C**" oder die Verschiebezahl "**2**", weil das Alphabet so um 2 Buchstaben verschoben wird, dass der Buchstabe "A" zum "C" hin verschoben wird:

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**
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**

Damit wird die geheime Nachricht verschlüsselt:

D	**i**	**e**	**s**	**e**		**N**	**a**	**c**	**h**	**r**	**i**	**c**	**h**	**t**		**i**	**s**	**t**		**g**	**e**	**h**	**e**	**i**	**m**	**!**
F	**k**	**g**	**u**	**g**		**P**	**c**	**e**	**j**	**t**	**k**	**e**	**j**	**v**		**k**	**u**	**v**		**i**	**g**	**j**	**g**	**k**	**o**	**!**

Zur Demonstration lässt sich auch gut das Wort **HALLO** in **JCNNQ** umwandeln, das Schlüsselwort hat als zweiten Buchstaben den Schlüsselbuchstaben (hier "C") und hat einen Buchstaben doppelt (eine Schwäche des Verfahrens).

☑ Anmerkung: Cäsar-entschlüsseln:
Um den verschlüsselten Text wieder zu entschlüsseln braucht man nur die folgende Verschiebezahl (26 - Verschiebezahl) verwenden und den verschlüsselten Text neu verschlüsseln.

Noch eine Anmerkung zu den Zahlen: Die Verschiebe-Zahl "0" entspricht der Verschiebung um 0 Buchstaben, also von "A" nach "A", die Verschiebe-Zahl "25" ist die höchst mögliche Zahl und entspricht einer Verschiebung von "A" nach "Z".

Jede Verschiebezahl wird hier mit "**modulo 26**" gerechnet.

☑ Anmerkung: Cäsar-knacken:

- Frage: Ist das Cäsar-Verfahren sicher?
- Antwort: Die Schüler erkennen, dass sie maximal 25 Versuche benötigen. Selbst diese 25 Versuche lassen sich noch verringern, wenn die Schüler sich den verschlüsselten Text genauer ansehen. So steht in der Nachricht Fkgug Pcejtkejv kuv igjgko! ein Wort kuv mit drei Buchstaben und so viele Worte (der, die, das, ist, bei, ...) gibt es in der deutschen Sprache nicht.

Anmerkung:
Die unterschiedlichen Häufigkeiten der einzelnen Buchstaben in der jeweiligen Landessprache:

Die Caesar-Methode gehört zu den **monoalphabetischen** Methoden, denn jeder Buchstabe des Alphabets wird stets zu demselben (Geheimtext-) Buchstaben chiffriert.

Längere derartig chiffrierte Texte können über die Häufigkeit der auftretenden Buchstaben entschlüsselt werden. Bei normalen Texten ist bekannt, wie häufig (im Mittel) die Buchstaben vorkommen. Für "normale" deutsche Texte gilt die Häufigkeitsverteilung:

Buchstabe	a	b	c	d	**e**	f	g	h	i	j	k	l	m

Häufigkeit in %	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
Buchstabe	n	o	p	q	r	s	t	u	v	w	x	y	z
Häufigkeit in %	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

Diese Häufigkeitsverteilung wird beim "Galgenraten" entsprechend ausgenutzt. Hier könnte man etwas über die Verwendung der einzelnen Buchstaben in den Landessprachen nachdenken, denn jede Landessprache hat ja ihr eigenes Profil der Häufigkeitsverteilung. Beim Cäsar-knacken könnte man im verschlüsselten Text nach dem häufigsten Buchstaben suchen, das wird dann wohl entschlüsselt ein "E" sein.

1.2.1. Cäsar verschlüsseln

Beim Cäsar-verschlüsseln müssen neben der Eingabe und Ausgabe lediglich zwei Schritte beachtet werden:

- Im Programm wird in einer Schleife vom ersten bis zum letzten Buchstaben jeder einzelne Buchstabe des Klartextes verändert.
- Da die Buchstaben (ASCII-Zeichen) mathematisch nicht selbst zu verändern sind, müssen die entsprechenden ASCII-Zahlen verschoben werden. Das Verfahren hat drei Schritte:
  - Der Buchstabe als einzelnes Zeichen des Textes wird in seine ASCII-Wert übersetzt.
  - Zum ASCII-Wert als Zahl wird die Verschiebungszahl addiert.
  - Der ASCII-Wert wird wieder in den entsprechenden Buchstabe zurück übersetzt.

  Wird beim zweiten Schritt der Wertebereich von 90 (= 'Z') überschritten, so wird bei 65 (= 'A') wieder angefangen. Analog für die Kleinbuchstaben.

1.2.2. Aufgabe: Cäsar verschlüsseln

☑ Eingabe:
  ☐ Dateiname der Originaldatei
  ☐ Key einlesen

☑ Ausgabe:
  ☐ Die verschlüsselte Originaldatei

☑ wichtig:
  ☐ Erstellen Sie für die eigentliche Ver/Entschlüsselung eine eigene Funktion, die den Orginaltext/Chiffretext und den Key als Input bekommt und den ver/entschlüsselten Text zurückgibt.

1.2.3. Aufgabe: Cäsar entschlüsseln

☑ Eingabe:
  ☐ Dateiname der Chiffredatei
  ☐ Key einlesen

☑ Ausgabe:
  ☐ Die entschlüsselte Originaldatei

☑ wichtig:
☐ Erstellen Sie für die eigentliche Ver/Entschlüsselung eine eigene Funktion, die den Orginaltext/Chiffretext und den Key als Input bekommt und den ver/entschlüsselten Text zurückgibt.

☑ Hinweis:
Es muss lediglich die Verschiebungszahl geändert werden. (Dekodierungszahl= 26 - Verschiebungszahl).

1.2.4. +Aufgabe: Cäsar knacken

☑ Aufgabe: (k_caesar.c)
Auch hier sind nur zwei Schritte zu beachten:

- Um den entscheidenden Teil des Programms Cäsar-entschlüsseln, der Schleife zum Verändern der einzelnen Buchstaben des verschlüsselten Textes, wird eine weitere Schleife von 1 bis 25 für die Verschiebungszahl herum gebaut. Nun werden alle 25 Möglichkeiten für den Code ausprobiert und der verschlüsselte Text damit umgewandelt.

  Der Code gilt als geknackt und der verschlüsselte Text als entschlüsselt, wenn bei einer der Möglichkeiten ein lesbarer Text entsteht.

So könnte der Ausdruck des Programmes aussehen:

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

----- N.N. -----

--------------------

Verschlüsselter Text: JCNNQ

Verschiebe-Buchstabe        ---             Entschluesselter Text

A                           ---             JCNNQ
B                           ---             IBMMP
C                           ---             HALLO
D                           ---             GZKKN
E                           ---             FYJJM
F                           ---             EXIIL
G                           ---             DWHHK
H                           ---             CVGGJ
I                           ---             BUFFI
J                           ---             ATEEH
K                           ---             ZSDDG
L                           ---             YRCCF
M                           ---             XQBBE
N                           ---             WPAAD
O                           ---             VOZZC
P                           ---             UNYYB
Q                           ---             TMXXA
R                           ---             SLWWZ
```

S	**---**	**RKVVY**
T	**---**	**QJUUX**
U	**---**	**PITTW**
V	**---**	**OHSSV**
W	**---**	**NGRRU**
X	**---**	**MFQQT**
Y	**---**	**LEPPS**
Z	**---**	**KDOOR**

1.3. Skytale-Verfahren

☑ Skytale-verschlüsseln:
Beim Skytale-Verfahren wird eine Tabelle erstellt, der Text (ggf. ergänzt durch =) wird zeilenweise eingelesen und spaltenweise ausgelesen.
Der "**Key**" besteht aus zwei Zahlen: der **Anzahl der Spalten und der Anzahl der Zeilen** der Tabelle.

Als Beispiel nehmen wir den Klartext **Diese Nachricht ist geheim====** und den Key 6-5 für 6 Spalten und 5 Zeilen.

D	**i**	**e**	**s**	**e**	
N	**a**	**c**	**h**	**r**	**i**
c	**h**	**t**		**i**	**s**
t		**g**	**e**	**h**	**e**
i	**m**	**=**	**=**	**=**	**=**

Die verschlüsselte Nachricht lautet damit **DNctiiah mectg=sh e=erih= ise=** .

☑ Skytale-entschlüsseln:
Zum Entschlüsseln wird der Text in die gleiche Tabelle einfach spaltenweise eingelesen und zeilenweise ausgelesen.

So wird aus der verschlüsselten Nachricht **DNctiiah mectg=sh e=erih= ise=** wieder der entschlüsselte Klartext **Diese Nachricht ist geheim====** .

1.3.1. Aufgabe: Skytale-Verschlüsseln

☑ Eingabe:
☐ Name der zu verschlüsselnden Datei:
☐ Anzahl der Zeilen
☐ Anzahl der Spalten
☑ Ausgabe:
☐ Die verschlüsselte Datei

☑ wichtig:
- ☐ Erstellen Sie für die eigentliche Ver/Entschlüsselung eine eigene Funktion, die den Orginaltext/Chiffretext und den Key als Input bekommt und den ver/entschlüsselten Text zurückgibt.

☑ Hinweis: hier in C
Das vorgeschlagene Beispielprogramm ist recht karg und noch nicht sicher gegen Fehleingaben, aber es zeigt den Grundgedanken.

- ☐ v_skytale.c

```
/**
 * v_skytale.c
 * N.N.
 * gcc v_skytale.c -o v_skytale.exe
 * ./v_skytale.exe <original.txt >skytale-kodiert.txt
 */

#include <stdio.h>

#define ZEILEN 5
#define SPALTEN 6

int main(){
     char tabelle[ZEILEN][SPALTEN];
     int ch;
     int i,j;
     int fertig=0;
     //wenn ende der eingabe (EOF) erreicht, wird fertig auf 1 gesetzt

     //printf("v_skytale.c (%d ZEILEN x %d SPALTEN): Text eingeben ->",
     //    ZEILEN, SPALTEN);

     //Einlesen in ZEILENxSPALTEN
     for (i=0; i< ZEILEN; i++){
          for (j=0; j< SPALTEN; j++) {
               if (fertig==0) //noch nicht EOF erreicht
                    tabelle[i][j]= fgetc(stdin);
               else
                    //weil EOF erreicht wurde, fgetc() nicht aufrufen,
                    // sondern Rest mit = auffüllen
                    tabelle[i][j]= '=';

               if (tabelle[i][j] == EOF){
                    tabelle[i][j]= '=';
                    fertig= 1;
               }
          }
     }

     //Ausgeben in SPALTENxZEILEN
     for (i=0; i< SPALTEN; i++){
          for (j=0; j< ZEILEN; j++) {
               fputc(tabelle[j][i], stdout);
```

```
            }
        }

        return 0;
}
```

1.3.2. Aufgabe: Skytale-Entschlüsseln

☑ Eingabe:
- ☐ Die mit skytale verschlüsselte Datei
- ☐ Die Anzahl der Zeilen
- ☐ Die Anzahl der Spalten

☑ Ausgabe:
- ☐ Die entschlüsselte Originaldatei

☑ wichtig:
- ☐ Erstellen Sie für die eigentliche Ver/Entschlüsselung eine eigene Funktion, die den Orginaltext/Chiffretext und den Key als Input bekommt und den ver/entschlüsselten Text zurückgibt.

1.3.3. +Aufgabe: Skytale-Knacken

Zwei Schritte sind also hier zu beachteten:

- Um den entscheidenden Teil des Programms Skytale-entschlüsseln, den beiden Schleifen zum Einlesen und Auslesen, wird eine weitere Schleife (eigentlich zwei, eine innere und eine äußere für die unbekannte Zeilenzahl und die unbekannte Spaltenzahl) herum gebaut. Nun werden alle Möglichkeiten für den Code ausprobiert und der verschlüsselte Text damit umgewandelt. Der Code gilt als geknackt und der verschlüsselte Text als entschlüsselt, wenn bei einer der Möglichkeiten ein lesbarer Text entsteht.

- Schwierig ist das Deklarieren der Variablen "Tabelle". Sie kann dynamisch je nach probierter Spalten- und Zeilenzahl deklariert werden. Sie kann aber auch statisch genügend groß deklariert werden, die Tabelle muss ja nicht vollständig genutzt werden.

Hier ist ein Programme ggf. mit zusätzlichen Erläuterungen:
☑ k_skytale.c

```
/**
 * k_skytale.c
 * N.N
 * gcc k_skytale.c -o k_skytale.exe
 * ./k_skytale.exe <skytale-kodiert.txt >original.txt
 */

#include <stdio.h>

//statisches Array: nehmen 10x10 als maximalen Wert an
#define ZEILEN 10
#define SPALTEN 10
```

```
int main(){
     char tabelle[ZEILEN*SPALTEN];
     int anz_text;
     int ch;
     int i,j;
     int anz_spalten;

     //Einlesen in ein 1 dim.-Array
     i=0;
     while ((ch= fgetc(stdin)) != EOF)
          tabelle[i++]=ch;

     //Merke die Anzahl der gelesenen zeichen
     anz_text=i;

     //versuche alle möglichen spalten
     for (anz_spalten=1; anz_spalten< SPALTEN; anz_spalten++){
               printf("\n... Zeilen=%d Spalten=%d ...\n",
                         anz_text/anz_spalten, anz_spalten);

               //Ausgeben in ZEILENxSPALTEN
               for (i=0; i< anz_spalten;i++)
                    for (j=i; j< anz_text; j= j+anz_spalten)
                         if (tabelle[j] != '=')
                              fputc(tabelle[j], stdout);


     }

     return 0;
}
```

1.4. Vigenere-Verfahren

[wischenär]

Das Vigenère-Verfahren ist eine Weiterentwicklung des Cäsar-Verfahrens, es verwendet jedoch statt eines einzigen Key-Buchstaben ein **Key-Wort** mit mehreren Buchstaben.

Damit wird ein Buchstabe des Klartextes nicht immer auf den gleichen Schlüssel-Buchstaben verschoben, das Verfahren ist also **polyalphabetisch**.

☑ Beispiel:
Wir nehmen als Beispiel den Satz **Diese Nachricht ist geheim!**, hier aber jetzt ohne Leerstellen und Satzzeichen und nur in großen Buchstaben: **DIESENACHRICHTISTGEHEIM**

Wir nehmen als Key-Wort **HALLO**

☑ Wir bauen folg. Tabelle

TEXT	D	I	E	S	E	N	A	C	H	R	I	C	H	T	I	S	T	G	E	H	E	I	M
KEY	H	A	L	L	O	H	A	L	L	O	H	A	L	L	O	H	A	L	L	O	H	A	L
CHIFFRE	K	I	P	D	S	U	A	N	S	F	P	C	S	E	W	Z	I	T	R	V	L	I	X

☑ Um das erste Zeichen z.B. zu verschlüsseln, geht man folgendermaßen vor:
Text: D wird als **Zeilenindex (hier 3)**
Key: H wird als **Spaltenindex** (hier 7)

Chiffre: 1.Zeichen → vigenereTabelle[D][H] → *K* (hier 10)
usw.

Zum Verschlüsseln benutzt man nun das sog. Vigenère-Quadrat:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	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
1	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
2	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
3	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
...																										
7	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
...																										
11	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
...																										
14	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
...																										
25	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

1.4.1. Vigenere ver/entschlüsseln

Wenn man anstatt der Zeichen den **Index in der Vigenere-Tabelle** verwendet, sieht man folgendes:

TEXT	D(3)	I(8)	E(4)	S(18)	...	
KEY	H(7)	A(0)	L(11)	L(11)		
CHIFFRE	K(10)	I(8)	P(15)	D(29) D(29%26) D(3)	...	

☑ Anmerkung:
statt des Verschiebebuchstaben wird hier mit Index die VerschiebeZahl (s.o. Caesar) gemeint.

A->0
B->1
C->2
D->3
...

Merke:
Beim Verschlüsseln geht man also folgenderm. vor:

```
        INDEX vom TEXT-Zeichen
+       INDEX vom KEY-Zeichen
        INDEX vom KODIERTEN Zeichen
=============================
```

Beim Entschlüsseln gilt:

```
        INDEX vom KODIERTEN Zeichen
-       INDEX vom KEY-Zeichen
        INDEX vom TEXT-Zeichen
=============================
```

d.h wir können mit

int indexText= toupper(text[i])-'A';
int indexKey= toupper(key[j])-'A'; // entspricht der VerschiebeZAHL

diesen Index berechnen und dann mit

char cipher= 'A' + (indexText + indexKey)%26;

das kodierte Zeichen ermitteln.

Hier nun das C-Programm

```
/*
 * vigenere.c
 * N.N
 */

#include <stdio.h>
#include <ctype.h>
#include <string.h>

void encryptVigenere(char* txt, const char* key);
void decryptVigenere(char* txt, const char* key);

#define STRLEN 256
int main(){
     int choice;
     char input[STRLEN];
     char key[STRLEN];

     while(1){
          printf("\n");
          printf("\n1. Encrypt Text\n");
          printf("2. Decrypt Text\n");
          printf("3. Exit\n");
```

```
            printf("Enter Your Choice : ");

            scanf("%d",&choice);
            fgetc(stdin); //wegen enter


            if(choice == 3)
                  exit(0);
            else if(choice == 1) {
                  printf("Text to be encrypted: ");
                  gets(input);

                  printf("Encryption Key: ");
                  gets(key);

                  encryptVigenere(input, key);

                  printf("Encrypted Text= <%s>\n", input);
            }
            else if(choice == 2){
                  printf("Text to be decrypted: ");
                  gets(input);

                  printf("Encryption Key: ");
                  gets(key);

                  decryptVigenere(input, key);

                  printf("Decrypted Text= <%s>\n", input);
            }
            else
                  printf("Please Enter Valid Option.");
      }
}



void encryptVigenere(char* txt, const char* key){
      int i; // index for txt
      int j; // index for key


      for(i=0,j=0; i<strlen(txt); i++,j++){

            //repeat the key if you are at end of it.
            if(j>=strlen(key)){
                  j=0;
            }

            //actual logic ->
            // character from input +
            // character from key % 26 is encrypted charater
            int indexText=  toupper(txt[i])-'A';
            int indexKey= toupper(key[j])-'A';
```

```
            int indexCipher= (indexText + indexKey) %26;
            char cipher= 'A' + indexCipher;

            txt[i]= cipher;
      }
}


void decryptVigenere(char* txt, const char* key){
      int i; // index for txt
      int j; // for key
      int value;


      for(i=0,j=0; i<strlen(txt); i++,j++){

            //repeat the key if you are at end of it.
            if(j>=strlen(key)){
                  j=0;
            }

            //similar to encipher only difference is you need to subtract
            value = (toupper(txt[i])-64)-(toupper(key[j])-64);

            //if value is negative.
            // We have to rotate it backwards (like backwards from z,y,x)
            // so add it to 26
            // (it's a negative value to adding will
            // actually cause subtraction)
            // to get original character.

            if( value < 0){
                  value = 26 + value;
            }
            txt[i]= 'A' + (value % 26);
      }
}
```

1.4.2. Aufgabe: vigenere verschlüsseln

☑ Eingabe:
- ☐ Name der zu verschlüsselnden Datei
- ☐ Eingabe des Key-Wortes

☑ Ausgabe:
- ☐ Die verschlüsselte Datei.

☑ wichtig:
- ☐ Erstellen Sie für die eigentliche Ver/Entschlüsselung eine eigene Funktion, die den Orginaltext/Chiffretext und den Key als Input bekommt und den ver/entschlüsselten Text zurückgibt.

1.4.3. Aufgabe: vigenere entschlüsseln

☑ Eingabe:
  ☐ Dateiname der Chiffredatei
  ☐ Keywort einlesen
☑ Ausgabe:
  ☐ Die entschlüsselte Originaldatei

☑ wichtig:
  ☐ Erstellen Sie für die eigentliche Ver/Entschlüsselung eine eigene Funktion, die den Orginaltext/Chiffretext und den Key als Input bekommt und den ver/entschlüsselten Text zurückgibt.

1.5. Weitere Aufgaben

1.5.1. Aufgabe: Arten der Verschlüsselungen

Suche im Internet nach einer Erklärung zu folgenden Begriffen:

1. symmetrische Verschlüsselung
2. asymmetrische Verschlüsselung
3. monoalphabetische Verschlüsselung
4. polyalphabetische Verschlüsselung

1.5.2. Aufgabe: SwingKrypto

Erstellen Sie ein Programm, das die Verwendung der in diesem Skript verwendeten Verschlüsselungsmethoden zum Ver- und Entschlüsseln ermöglicht.