Einführung in die Programmierung für Studierende der Naturwissenschaften
SS 2021

https://aam.uni-freiburg.de/agdo/lehre/ss21/prog/index.html

UNI FREIBURG

Codebreaker für Caesar-Verschlüsselung mit Häufigkeitsanalyse

von Daniel Rath und Theresa Maurer

Verschlüsselung und Entschlüsselung

Die Caesar-Verschlüsselung ist eine einfache, monoalphabetische Verschlüsselung. Die Verschlüsselung der Buchstaben a - z entspricht einer Verschiebung des Alphabets um beliebig viele Stellen. Zwischen Groß- und Kleinbuchstaben wird nicht unterschieden.[Beu07]
Beispielsweise wird bei einer Verschiebung um drei Buchstaben das a auf das d abgebildet, das b wird auf das e abgebildet, das c wird auf das f abgebildet, für die anderen Buchstaben wird diese Verfahrensweise analog weitergeführt. Nach der Systematik der Chiffrierung wird die Verschiebung nach dem z beim a weitergeführt. Also wird das w auf das z abgebildet und das x folglich auf das a.

Klartext:	a	b	c	d	e	f	...	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Chiffre:	d	e	f	g	h	i	...	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c

Tabelle 1: Beispiel für eine Caesar-Verschlüsselung mit Verschiebung um 3 Buchstaben

Die Verschlüsselung eines Textes lässt sich bei der Programmierung leichter umsetzen, wenn jeder Buchstabe in seiner ASCII-Darstellung betrachtet wird. Das heißt jeder Buchstabe wird als Zahl dargestellt, die Verschiebung entspricht einer Addition. Eine Verschiebung um drei Buchstaben entspricht einer Addition mit der Zahl 3.

Jedem Buchstaben des Alphabets ist eine Zahl im ASCII-Code zugeordnet:

Buchstabe	**ASCII**	**Buchstabe**	**ASCII**	**Zahl**	**ASCII**
A	65	a	97	0	48
B	66	b	98	1	49
C	67	c	99	2	50
⋮	⋮	⋮	⋮	⋮	⋮
J	74	j	106	9	57
⋮	⋮	⋮	⋮		
X	88	x	120		
Y	89	y	121		
Z	90	z	122		

Tabelle 2: Groß-/Kleinbuchstaben und Zahlen mit dem entsprechenden ASCII-Code [Roi93]

Der Einfachheit halber beschränken wir uns hier auf die Kleinbuchstaben von a-z, also die ASCII-Zeichen 97-122. Ein Kleinbuchstabe x in ASCII-Darstellung kann nun durch eine additive Funktion

$$C(x) = ((x - 97) + c) \bmod 26) + 97$$

verschlüsselt werden, für ein $c \in \mathbb{Z}$. Der verschlüsselte Buchstabe ist wieder eine Zahl zwischen 97-122 und somit auch ein Kleinbuchstabe. Das c bewirkt die Verschiebung des Alphabets.
Die zugehörige Dechiffrierfunktion ist

$$D(x) = ((x - 97) - c) \bmod 26) + 97\dot{t}$$

da gilt

$$\begin{aligned} D(C(x)) &= ((((x - 97) + c) \bmod 26) + 97 - 97) - c) \bmod 26) + 97 \\ &= (((x - 97) + c) - c) \bmod 26) + 97 \\ &= (((x - 97) \bmod 26) + 97 = x, \text{ für } 97 \leq x \leq 122 \end{aligned}$$

Für ein c, das die Funktionen C(x) und D(x) eindeutig bestimmt, bewirkt jede Zahl c', mit der Eigenschaft $c = c' \bmod 26$, dieselbe Verschiebung des Alphabets. Daher gibt es für die Chiffrier-/Dechiffrierfunktion nur 26 verschiedene Möglichkeiten. Somit sind Caesar-Verschlüssungen sehr einfach zu knacken und stellen in der Anwendung keine sichere Chiffrierung da. [Oll03]

Entschlüsselung eines Caesar-chiffrierten Textes durch Häufigkeitsanalyse

Um einen Caesar-verschlüsselten Text zu entschlüsseln, muss die Dechiffrierfunktion D(x) gefunden werden und auf jeden Buchstaben angewendet werden. Dazu reicht aus, die Verschiebung c zu finden, da durch das c die Funktion D(x) eindeutig bestimmt wird.
In der deutschen Sprache kommt das e eindeutig am häufigsten vor. Durch eine Häufigkeitsanalyse der Buchstaben des verschlüsselten Textes, lässt sich so der Buchstabe y (in Ascii-Darstellung), auf den das e abgebildet wurde, ermitteln. Es gilt $D(y) = ((y - 97) - c) \bmod 26) + 97 = 101$ und somit $c = y - 101$.

Buchstabe	**Häufigkeit in %**	**Buchstabe**	**Häufigkeit in %**
a	6.51	n	9.78
b	1.89	o	2.51
c	3.06	p	0.79
d	5.08	q	0.02
e	17.40	r	7.00
f	1.66	s	7.27
g	3.01	t	6.15
h	4.76	u	4.35
i	7.55	v	0.67
j	0.27	w	1.89
k	1.21	x	0.03
l	3.44	y	0.04
m	2.53	z	1.13

Tabelle 3: Häufigkeit der Buchstaben in der deutschen Sprache [Lin12]

Dabei ist zu beachten, dass die Häufigkeitsanalyse aus mathematischen Gründen nur bei ausreichend langen Texten sicher funktioniert. In unserem Programm benötigt der eingegebene Text deshalb mindestens 26 Buchstaben.

C++-Programm zur Ent-/Verschlüsselung

Die Verschlüsselungsfunktion: encrypt_text() in der Datei encrypt.cc

Die Funktion nimmt als Argument einen zu verschlüsselnden Text, wobei der eingegebene Text mindestens 25 Zeichen enthalten muss, damit sich der Text auch wieder mit einer Häufigkeitsanalyse entschlüsseln lässt. Dann muss noch der Kleinbuchstabe eingegeben werden, auf den das a bei der Verschlüsselung abgebildet werden soll, um die Differenz c zu bestimmen.
Anschließend werden alle Großbuchstaben des Textes in Kleinbuchstaben umgewandelt. Dann wird über den ganzen Text iteriert und auf jeden Buchstaben die Verschlüsselungsfunktion C(x) angewandt, wobei Sonderzeichen nicht verschlüsselt werden, sondern einfach an der richtigen Stelle im Text wieder eingefügt werden.
Der verschlüsselte Text wird dann als string zurückgegeben.

Die Entschlüsselungsfunktion: decrypt_text() in der Datei decrypt.cc

Die Funktion nimmt einen Caesar-verschlüsselten Text als Eingabe, der mindestens die Länge 26 hat. Dann wird die Häufigkeit jedes Buchstaben im Text gezählt und in eine Priority Queue eingefügt, wobei der key die absolute Häufigkeit als integer ist und der value der entsprechende Buchstabe.
Der am häufigsten verwendete Buchstabe im verschlüsselten Text entspricht dann dem value des obersten Elements in der Priority Queue und kann einfach ausgelesen werden. Daraus kann die Differenz c wie oben beschrieben berechnet werden und anschließend die Funktion D(x) auf den gesamten Text angewendet werden.
Am Ende werden die ersten 30 Buchstaben des entschlüsselten Textes ausgegeben und überprüft, ob das wirklich der entschlüsselte Text ist. Falls nicht, wird angenommen, dass der nächst häufigste Buchstabe das e war und der Text wird damit entschlüsselt. Das wird dreimal wiederholt, wenn dann kein richtiger Text gefunden wurde, kann angenommen werden, dass der eingegebene Text gar nicht Caesar-verschlüsselt war.
Entsprechen die ersten 30 Buchstaben einem korrekten Text, wird der ganze entschlüsselte Text als string zurückgegeben.

Demonstrationsfunktion: demo_module() in der Datei demonstration.cc

Die Funktion demo_module nimmt einen Text als Eingabe und zeigt einmal die Verschlüsselung und Entschlüsselung. Die Funktion dient nur zur einfacheren Präsentation des Programms.
Auf den Text wird einmal die Funktion encrypt_text angewendet und der verschlüsselte Text ausgegeben. Dann wird auf den verschlüsselten Text die Entschlüsselungsfunktion decrypt_text angewendet und der Text wieder ausgegeben.

Formatierungsfunktion: format_for_print() in der Datei format_print.cc

Durch die Funktion format_for_print kann die Ausgabe von Texten in der Konsole schöner dargestellt werden. Das erleichtert die Nutzung auch bei kleineren Konsolenfenstern.

main_program.cc

Das main_programm ist als einzige Datei zur Ausführung gedacht. Beim Ausführen wird abgefragt, welche der vier Optionen Verschlüsseln, Entschlüsseln eines Textes, Demonstration der Funktionen oder Beenden des Programms gewählt werden soll. Damit wird hier der gesamte Programmablauf koordiniert, die eigentliche Logik passiert aber an einer anderen Stelle.

Tabellenverzeichnis

Literatur

[Beu07] Albrecht Beutelspacher. „Cäsar oder Aller Anfang ist leicht!“ In: *Kryptologie*. Vieweg, 2007, S. 1–25. DOI: 10.1007/978-3-8348-9209-6_1.

[Lin12] Alex Lindner. *Geheimschrift und Codes » Kryptographie*. Zugriff am 25.07.2021. Okt. 2012. URL: https://www.seo-woman.de/bilder/2012/buchstaben-haeufigkeit-deutsches-alphabet.jpg.

[Oll03] Jan Olligs. „Mathematische Methoden der Analyse einfacher Kryptosysteme“. In: *Kryptosysteme* (2003), S. 3–7.

[Roi93] Roitzsch. „COBOL - Das Handbuch für den professionellen Programmierer : Auf der Basis des ANSI-Standards unter Berücksichtigung der IBM-Erweiterungen unter VS COBOL II“. In: Wiesbaden: Vieweg+Teubner Verlag, 1993, S. 595–597. ISBN: 9783322878021.