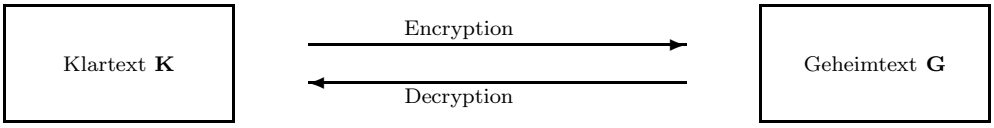
Einführung

Um einen Einstieg in die Kryptographie zu erhalten, ist es notwendig, verschiedene Begrifflichkeiten zu unterscheiden. Die Kryptographie als solches bezeichnet eine Wissenschaftsdisziplin, deren Inhalt es ist, Verschlüsselungsverfahren zu entwickeln, die primär dem Schutz geheimer Daten dienen. Der zu verschlüsselnde Text wird dabei als Klartext bezeichnet, der verschlüsselte Text im Weiteren als Geheimtext. Im Laufe der Jahrhunderte war es notwendig Botschaften über offene bzw. unsichere Kanäle auszutauschen. Während früher der Transport von beschriebenem Papier den Kanal, darstellte ist heute der Austausch über elektronische Mail möglich. Das grundlegende Problem ist dabei jedoch das gleiche geblieben. Es gilt, die Daten zu sichern.



Damit lassen sich Verschlüsselungsverfahren (Kryptosysteme) generell auch funktional erfassen.

$$E(K) = G$$

$$D(G) = K$$

$$D(E(K)) = K$$

Die Funktionen sind damit injektiv, es gehen im Idealfall keine Daten verloren. Verwendung finden kryptographische Verfahren heute in den verschiedensten Bereichen, wobei die Geheimhaltung der Nachricht nur eine wesentliche Aufgabe dieser Verfahren ist. Wichtig ist darüber hinaus die Eindeutigkeit der Identität des Absenders und die daraus resultierende Verbindlichkeit des Absenders für den Inhalt der Nachricht.

Die Rolle des Schlüssels

Ein gutes Kryptosystem muss verschiedenen Anforderungen genügen. So sollte zum Beispiel die Ver - und Entschlüsselung einfach sein und den Verfasser bzw. Empfänger nicht vor unlösbare Aufgaben stellen. Der verschlüsselte Text sollte nicht wesentlich länger sein, als die zu verschlüsselnde Nachricht, da sonst die Übertragungszeit erheblich ausgedehnt würde. Die wichtigste Eigenschaft ist jedoch die Unmöglichkeit einer Decodierung. Gerade hier gibt es jedoch durch die fortlaufende Entwicklung im Soft - und Hardwarebereich Veränderungen.

Aktuell sind sogenannte `Public - Key Verfahren` bedeutsam. Die Funktionalität dieser Verfahren basiert auf dem sogenannten `Kerhoffschen Prinzip`, welches besagt, dass der Algorithmus zur Verschlüsselung veröffentlicht werden muss, der Schlüssel selbst aber geheim bleibt. Die Forderung nach Veröffentlichung leitet sich aus der Notwendigkeit ab,

die Sicherheit der Kryptosysteme ständig zu testen. Unterbleibt die Veröffentlichung, ergeben sich gravierende Nachteile. Aktuelle Beispiele belegen die Richtigkeit des Kerhoffschen Prinzips.

Die Funktionalität der Public - Key Verfahrens basiert auf der Veröffentlichung des Codierungsschlüssels. Jeder Nutzer ist in der Lage diesen Schlüssel zu nutzen. Dazu ein Beispiel.

PGP - Pretty Good Privacy

Alice stellt ihren öffentlichen Schlüssel zur Verfügung, zum Beipiel als Anhang der Mailadresse. Den geheimen Schlüssel behält Alice für sich (zum Beipiel auf einem Stick). Bob lädt sich den öffentlichen Schlüssel herunter, verschlüsselt seine Nachricht und schickt diese an Alice. Alice kann nun mit Hilfe ihres geheimen Schlüssels die Nachricht von Bob entschlüsseln. Stellt Bob seinen öffentlichen Schlüssel ins Netz hat Alice die gleiche Chance, Bob eine Nachricht zu senden.

Unter dem Stichwort PGP (Pretty Good Privacy) wird dieses Verfahren für E- Mail bereits seit Jahren erfolgreich genutzt. Ein zugrundeliegendes Kryptosystem ist das sogenannte RSA - Verfahren, auf welches an späterer Stelle eingegangen wird.

Klassische Chiffren

Die Entwicklung moderner Verschlüsselungsverfahren basiert auf einer langen Geschichte. Entwickelte Kryptosysteme wurden jedoch immer wieder durch eine geeignete Kryptoanalyse in Frage gestellt. Programmiertechnisch nachvollziehen läßt sich die Kryptoanalyse am ehesten bei den sogenannten klassischen Verfahren. Gemeint sind hierbei Verfahren, die bis in die 50ger Jahre entwickelt wurden. Eine vergleichsweise große Gruppe dieser Chiffren stellen die sogenannten `Substitutionschiffren` dar. Diese sind dadurch gekennzeichnet, dass die Klartextzeichen über einer Alphabetmenge durch andere Zeichen ersetzt werden, ohne die Position zu ändern.
Wird das Klartextzeichen dabei immer auf ein und dasselbe Geheimtextzeichen abgebildet, spricht man von sogenannten **monoalphabetischen Chiffren**, ist diese Abbildung nicht injektiv, verwendet man den Begriff **polyalphabetisch**.

Der Caesarcode

Der Caesarcode, benannt nach seinem Erfinder Julius Caesar (100 bis 44 v. Chr), ist ein Verschiebechiffre und zählt danmit zur Gruppe der monoalphabetischen Substitutionschiffren. Das Prinzip dieses Kryptosystems ist relativ einfach. Jedes Klartextzeichen wird durch ein um den Index i verschobenes Zeichen ersetzt. Da die zugrundeliegende Alphabetmenge eine entsprechende Mächtigkeit von z.B. $M = |26|$ besitzt, wird bei einem Überlauf der Restklassenoperator verwendet.

Daraus ergibt sich der grundlegende Zusammenhang : $k \mapsto (k + i)\ mod\ M$

Caesar in Python

Eine Implementierung in der Sprache Python stellt sich wie folgt dar. Analysieren Sie die Klasse und entwerfen Sie die Methoden `code` und `decode`, mit welchen ein durch Caesar codierter Text unter Kenntnis der Verschiebungsweite codiert und decodiert werden kann. Testen Sie ihre Klasse in der vorgegebenen Testumgebung.

```
1  class caesar :
2
3    def __init__(self):
4
5        self.schub = 0
6
7    def eingabe (self) :
8
9        a = raw_input("String␣>")
10       return a
11
12   def verschiebung (self, verschiebung ):
13
14       self.schub = verschiebung
15
16   def caesar_code (self, orginal):
17
18       # TODO
19
20   def caesar_decode (self, bild):
21
22       # TODO
23
24 if __name__=="__main__":
25
26
27    Caesar = caesar ()
28    Caesar.verschiebung (3)
29    a = Caesar.eingabe ()
30    b = Caesar.caesar_code (a)
31    c = Caesar.caesar_decode (b)
32
33    print b
34    print c
```

Das Verfahren ist jedoch in keiner Weise sicher. Da die Buchstaben in deutschen Alphabeten mit einer unterschiedlichen Häufigkeit auftreten, ist eine Umwandlung dieses Verschiebechiffre möglich. Die Weite der Verschiebung ist bereits durch einen Buchstaben gegeben. Im Internet und in der Literatur sind Häufigkeitstabellen zur Buchstabenhäufigkeit zu finden. In der Regel ist eine Erfassung des Buchstaben e (prozentuale Häufigkeit : 17,4 %) ausreichend. Der zweithäufigste Buchstabe n muss selten bemüht werden. Ist der Text sehr kurz, ist ein differenzierteres Vorgehen notwendig. Im schlimmsten Fall ist die Häufigkeit von Paaren (z.B. en) heranzuziehen, aber auch diese ist hinlänglich statistisch erfasst. Die Kryptoanalyse läßt sich in diesem Fall über ein einfaches Histogramm erfassen. Eine in Python mögliche Implementierung, die nicht auf `built-in` Funktionen basiert wäre durch die Verwendung eines Dictionarys möglich.

Ein weiterer Typ der Substitutionschiffren sind sogenannte multiplikative Chiffren Hierbei wird ähnlich dem Caesarcode verfahren, allerdings sind die mathematischen Grundlagen komplexer. Die Verschiebung um eine feste Größe wird hier durch einen Multiplikator ersetzt.

Daraus ergibt sich der grundlegende Zusammenhang : $k \mapsto (k \cdot i)\ mod\ M$

Mathematische Grundlagen

Ähnlich dem Verschiebechiffre ist es relativ einfach, eine Codierfunktion zu generieren. Allerdings kann nicht jeder Faktor verwendet werden. Beispiele zeigen, dass für einige Faktoren die Funktion zwar eindeutig, aber nicht injektiv ist. Daraus ergibt sich die Problematik, dass nur für bestimmte Faktoren eine Dechiffrierfunktion existiert. Das Kriterium zur Bestimmung des erforderlichen Faktor läßt sich über den `ggT` aus Multiplikator und Mächtigkeit der Alphabetmenge bestimmen. Ist dieser größte gemeinsame Teiler 1 ist die Abbildung injektiv und es existiert zur vorhandenen Codierfunktion eine Dechiffrierfunktion. Ist diese Bedingung erfüllt, kann der inverse Multiplikator oder das `multiplikativ Inverse` der Funktion bestimmt werden. Der Algorithmus zur Bestimmung des multiplikativ Inversen erklärt sich wie folgt.

Euklidischer und erweiterter euklidischer Algorithmus

Geht man von einer Alphabetmächtigkeit von 26 aus und dem Multiplikator 7 läßt sich die Bedingung ggT (7, 26) relativ schnell nachweisen.

$ggt(7, 26)$

$\rightsquigarrow 26 = 3 \cdot 7\ \ R\ \ 5$

$\rightsquigarrow 7\ = 1 \cdot 5\ \ R\ \ 2$

$\rightsquigarrow 5\ = 2 \cdot 2\ \ R\ \ 1$

$\rightsquigarrow 1\ = 1$

Rückwirkendes Einsetzen liefert nach dem **Erweiterten Euklidischen Algorithmus** das sogenannte multiplikative Inverse.

$$
\begin{aligned}
1 = \quad & 5 - 2 \cdot 2 \\
& 5 - 2 \cdot (7 - 1 \cdot 5) && = -2 \cdot 7 + 3 \cdot 5 \\
& -2 \cdot 7 + 3 \cdot (26 - 3 \cdot 7) && = -11 \cdot 7 + 3 \cdot 26
\end{aligned}
$$

Damit ist - 11 das **multiplikativ Inverse** zu 7 und kann für die Dechiffrierfunktion verwendet werden. Eine mögliche Implementierung wäre die folgende, welche den rekursiven Abstieg des Euklidischen Algorithmus (ggT - Ermittlung) und den anschließenden Aufstieg ausgehend von einer Terminationsbedingung gut und knapp darstellt. Beim Aufstieg wird nun der Erweiterte Euklidische Algorithmus genutzt.

Der Algorithmus liefert ein Tupel. Aus diesem Grunde ist beim implementieren der entsprechenden Klasse das Inverse über den Index 1 einzubinden (siehe Beispielcode).

```
1 def eeuklid (a,b,d=0,x=0,y=0):
2
3     if b == 0 : return (a,1,0)
4     (d,x,y) = eeuklid (b, a%b)
5     return (d,y,x-(a//b)*y) # Ganzzahldivision
6
7
8 print eeuklid (7,26) [1]
```