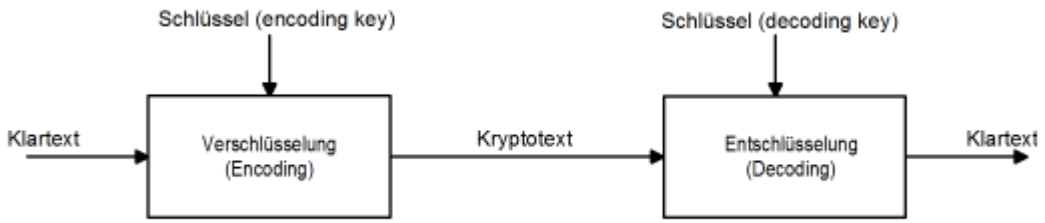
10.5 KRYPTOSYSTEME

EINFÜHRUNG

Das Geheimhalteprinzip spielt in unserer modernen Welt eine immer wichtigere Rolle. Zum Schutz de Privatsphäre, aber auch zur Geheimhaltung wichtiger staatlicher, industrieller und militärische Informationen ist es nötig, Daten derart zu verschlüsseln, dass die verschlüsselten Daten wohl in die Händ von Unbefugten fallen können, es aber ohne Bekanntgabe der Entschlüsselungsmethode unmöglich ode zumindest sehr schwierig ist, die Originalinformation herauszufinden.

Bei der **Verschlüsselung** (**encoding**) werden die Originaldaten in verschlüsselte (chiffrierte) Date umgewandelt. Bei der **Entschlüsselung** (**decoding**) werden die Originaldaten wieder hergestell Verwendet man für die Daten das Buchstabenalphabet, so spricht man auch von **Klartext** und **Kryptotext**.

Die Beschreibung des Verfahrens zum Entschlüsseln wird ***Schlüssel*** genannt. Es kann sich auch nur u eine einzige Zahl, eine Zahlen- oder eine Buchstabenfolge (ein Schlüsselwort) handeln. Wird bei Verschlüsseln und Entschlüsseln derselbe Schlüssel verwendet, so spricht man von einer **symmetrischen Kryptoverfahren**, andernfalls von einem **asymmetrischen Kryptoverfahren**.



PROGRAMMIERKONZEPTE:*Verschlüsselung, Entschlüsselung, Symmetrisches/Asymmetrisches Kryptoverfahren, Caesar-, Vigenère-, RSA-Verschlüsselung, Privater/Öffentlicher Schlüssel*

CAESAR-VERSCHLÜSSELUNG

Nach der Überlieferung soll bereits Julius Cäsar (100 v.Chr. - 44 v.Chr.) folgendes Verfahren für sein militärische Korrespondenz angewendet haben: Jeder Buchstabe des Klartexts wird dabei alphabetisch u eine bestimmte feste Schlüsselzahl nach rechts verschoben, wobei man nach Z wieder bei A weiterfährt.

Das Alphabet wird bei dieser Methode also in einem **Ringbuffer** angelegt. Mit dem Schlüssel 3 werden die Buchstaben A in D, B in E, C in F, D in G, usw. verschlüsselt.

In deinem Programm verwendest du Textdateien für die Daten, damit du sie leicht verändern und weitergeben kannst. Den Klartext schreibst du mit irgendeinem Texteditor in die Datei *original.txt*, die du im Verzeichnis, in dem sich dein Programm befindet, speichern musst. Verwende für den Text nur Grossbuchstaben und das Leerzeichen. Du kannst mehrere Zeilen schreiben, also beispielsweise

```
HEUTE TREFFEN WIR UNS UM ACHT
LIEBER GRUSS
TANIA
```

Der Encoder verschlüsselt den von der Datei eingelesene Textstring *msg* mit der Funktion *encode(msg* wobei ausser für den Zeilenumbruch \n jedes Zeichen durch das entsprechende Kryptozeichen ersetzt wir

```
import string
key = 4
alphabet = string.ascii_uppercase + " "

def encode(text):
    enc = ""
    for ch in text:
        if ch != "\n":
            i = alphabet.index(ch)
            ch = alphabet[(i + key) % 27]
        enc += ch
    return enc

fInp = open("original.txt")
text = fInp.read()
fInp.close()

print "Original:\n", text
krypto = encode(text)
print "Krypto:\n", krypto

fOut = open("secret.txt", "w")
for ch in krypto:
    fOut.write(ch)
fOut.close()
```

Programmcode markieren (Ctrl+C kopieren, Ctrl+V einfügen)

Dein verschlüsselter Text sieht wie folgt aus:

```
LIYXIDXVIJJIRD MVDYRWDYQDEGLX
PMIFIVDKVYWW
XERME
```

Der Decoder ist ganz analog aufgebaut, nur dass die Zeichen im Alphabet rückwärts verschoben werden.

```
import string
key = 4
alphabet = string.ascii_uppercase + " "

def decode(text):
    dec = ""
    for ch in text:
        if ch != "\n":
            i = alphabet.index(ch)
            ch = alphabet[(i - key) % 27]
        dec += ch
    return dec

fInp = open("secret.txt")
krypto = fInp.read()
fInp.close()

print "Krypto:\n", krypto
msg = decode(krypto)
print "Message:\n", msg

fOut = open("message.txt", "w")
for ch in msg:
    fOut.write(ch)
fOut.close()
```

Programmcode markieren (Ctrl+C kopieren, Ctrl+V einfügen)

MEMO

Beachte, dass du im Kryptotext auch alle Leerzeichen beibehalten musst, auch wenn diese am Anfan oder am Ende einer Zeile stehen. Es ist klar, dass die Verschlüsselung leicht geknackt werden kann. E genügt beispielsweise, die Schlüsselzahlen 1..26 auszuprobieren

VERSCHLÜSSELUNG NACH DEM VIGENÈRE-VERFAHREN

Du kannst die Caesar-Verschlüsselung sicherer machen, indem du auf jedes Zeichen des Klartexts eine unterschiedliche alphabetische Verschiebung anwendest. Diese sogenannte polyalphabetische Substitution könnte als Schlüssel irgendeine Permutation von 27 Zahlen verwenden. Davon gibt es eine riesige Zahl, nämlich

$27! = 10'888'869'450'418'352'160'768'000'000 \approx 10^{27}$

Etwas einfacher ist die Verwendung eines Schlüsselworts, dem die Liste der entsprechenden Zeichen im Alphabet zugeordnet ist, also beispielsweise dem Schlüssel ALICE die Liste [0, 11, 8, 2, 4]. Bei der Verschlüsselung werden dann die Zeichen der Reihe nach um 0, 11, 8, 2, 4 und dann wiederholt um 0, 11,... Zeichen alphabetisch verschoben.

Blaise Vigenère (1523-1596)

```
import string
key = "ALICE"
alphabet = string.ascii_uppercase + " "

def encode(text):
    keyList = []
    for ch in key:
        i = alphabet.index(ch)
        keyList.append(i)
    print "keyList:", keyList
    enc = ""
    for n in range(len(text)):
        ch = text[n]
        if ch != "\n":
            i = alphabet.index(ch)
            k = n % len(key)
            ch = alphabet[(i + keyList[k]) % 27]
        enc += ch
    return enc

fInp = open("original.txt")
text = fInp.read()
fInp.close()

print "Original:\n", text
krypto = encode(text)
print "Krypto:\n", krypto

fOut = open("secret.txt", "w")
for ch in krypto:
    fOut.write(ch)
fOut.close()
```

Programmcode markieren (Ctrl+C kopieren, Ctrl+V einfügen)

Der Decoder ist wiederum praktisch identisch.

```
import string
key = "ALICE"
alphabet = string.ascii_uppercase + " "

def decode(text):
    keyList = []
    for ch in key:
        i = alphabet.index(ch)
        keyList.append(i)
    print "keyList:", keyList
    enc = ""
    for n in range(len(text)):
        ch = text[n]
        if ch != "\n":
            i = alphabet.index(ch)
            k = n % len(key)
            ch = alphabet[(i - keyList[k]) % 27]
        enc += ch
    return enc

fInp = open("secret.txt")
krypto = fInp.read()
fInp.close()

print "Krypto:\n", krypto
msg = decode(krypto)
print "Message:\n", msg

fOut = open("message.txt", "w")
for ch in msg:
    fOut.write(ch)
fOut.close()
```

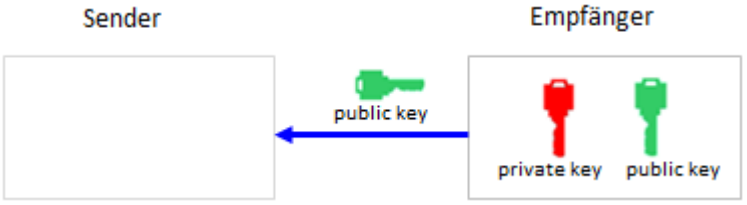
Programmcode markieren (Ctrl+C kopieren, Ctrl+V einfügen)

MEMO

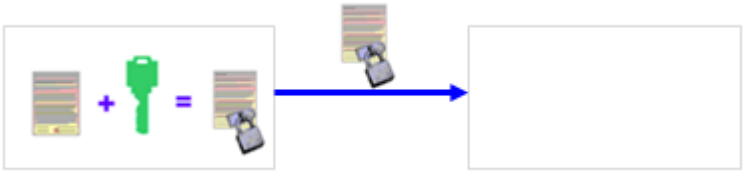
Die Vigenère-Verschlüsselung wurde bereits im 16. Jh. von Blaise de Vigenère erfunden und ga Jahrhunderte lang als sehr sicher. Kennt man die Länge 5 des Schlüsselworts, so muss man immerhi noch 26^5 = 11'881'376 Schlüsselzahlen durchprobieren, ausser man weiss etwas über das verwendet Wort, beispielsweise, dass es sich um den Vornamen einer Frau handelt.

VERSCHLÜSSELUNG NACH DEM RSA-VERFAHREN

Bei diesem Verfahren, das nach ihren Erfindern Rivest, Shamir und Adleman benannt ist, wird ei Schlüsselpaar verwendet, nämlich ein privater (private key) und ein öffentlicher Schlüssel (public key). Di Originaldaten werden mit dem öffentlichen Schlüssel verschlüsselt und mit dem privaten Schlüss entschlüsselt. Es handelt sich also um ein asymmetrisches Kryptoverfahren.



Schritt 1:
Der Empfänger erzeugt den *privat key* und den *public key* und schic den *public key* an den Sender.



Schritt 2:
Der Sender verschlüsselt sein Nachricht mit dem *public key* ur schickt den verschlüsselten Te zurück.

Schritt 3:
Der Empfänger entschlüsselt de Text mit dem *private key*.

Die Schlüssel werden mit folgendem Algorithmus erzeugt, der auf der Zahlentheorie beruht [mehr...].

Zuerst werden zwei Primzahlen p und q gewählt, die für ein sicheres System mehrere hundert Stelle haben sollten. Man multipliziert diese und bildet m = p*q. Aus der Zahlentheorie weiss man, dass di Eulersche Funktion φ(m) = (p-1)*(q-1) die Zahl der teilerfremden Zahlen zu m ist (a, b sind teilerfrem wenn der grösste gemeinsame Teiler ggT(a,b) = 1 ist).

Als nächstes wählt man eine Zahl e, die kleiner als φ und teilerfremd zu φ ist. Damit ist der öffentlich Schlüssel bereits erstellt, er besteht aus dem Zahlenpaar:

Öffentlicher Schlüssel: [m, e]

Hier ein Beispiel mit den kleinen Primzahlen p = 73 und q = 151:

m = 73 * 151 = 11023, φ = 72 * 150 = 10800, e = 11 (gewählt teilerfremd zu φ)

Öffentlicher Schlüssel: [m, e] = [11023, 11]

Der private Schlüssel besteht dann aus dem Zahlenpaar:

Privater Schlüssel: [m, d]

wobei für die Zahl d muss gelten: (d * e) mod φ = 1

(da e und φ teilerfremd sind, sagt das Lemma von Bézout aus der Zahlentheorie, dass die Gleichun mindestens eine Lösung hat).

Du kannst mit deinen Werten für e und φ die Zahl d mit einem einfachen Programm bestimmen, indem d in einer for-Schleife 100 000 Werte für d ausprobierst.

```
e = 11
phi = 10800

for d in range(100000):
    if  (d * e) % phi == 1:
        print "d", d
```

Du erhältst mehrere Lösungen (5891, 16691, 27491, 49091, usw.). Im Prinzip brauchst du aber nur di erste, um den privaten Schlüssel festzulegen.

Privater Schlüssel: [m, d] = [11023, 5891]

Die Berechnung des privaten Schlüssels ist hier nur deswegen so einfach, da du die Zahlen p und q un damit auch die Zahl φ kennst. Ohne Kenntnis dieser Zahlen lässt sich der private Schlüssel nur mit einer

grossen Aufwand berechnen

Mit dem RSA-Algorithmus werden **Zahlen** verschlüsselt. Um einen Text zu verschlüsseln, verwendest d den ASCII-Code jedes Zeichens und bildest mit dem öffentlichen Schlüssel [m, e] de Verschlüsselungswert s für den Geheimtext gemäss der Formel

$s = r^e$ (modulo m).

Diese Verschlüsselungszahlen schreibst du zeilenweise in die Datei *secret.txt*.

```
publicKey = [11023, 11]

def encode(text):
    m = publicKey[0]
    e = publicKey[1]
    enc = ""
    for ch in text:
        r = ord(ch)
        s = int(r**e % m)
        enc += str(s) + "\n"
    return enc

fInp = open("original.txt")
text = fInp.read()
fInp.close()

print "Original:\n", text
krypto = encode(text)
print "Krypto:\n", krypto

fOut = open("secret.txt", "w")
for ch in krypto:
    fOut.write(ch)
fOut.close()
```

Programmcode markieren (Ctrl+C kopieren, Ctrl+V einfügen)

Im Decoder liest du die Zahlen von der Datei *secret.txt* zuerst in eine Liste. Zum Entschlüsseln berechnes du mit dem privaten Schlüssel aus der Verschlüsselungszahl s die ursprüngliche Zahl gemäss der Formel

$r = s^d$ (modulo m).

Dies ist der ASCII-Code des ursprünglichen Zeichens.

```
privateKey = [11023, 5891]

def decode(li):
    m = privateKey[0]
    d = privateKey[1]
    enc = ""
    for c in li:
        s = int(c)
        r = s**d % m
        enc += chr(r)
    return enc

fInp = open("secret.txt")
krypto = []
while True:
   line = fInp.readline().rstrip("\n")
   if line == "":
       break
```

```
   krypto.append(line)
fInp.close()

print "Krypto:\n", krypto
msg = decode(krypto)
print "Message:\n", msg

fOut = open("message.txt", "w")
for ch in msg:
    fOut.write(ch)
fOut.close()
```

Programmcode markieren (Ctrl+C kopieren, Ctrl+V einfügen)

MEMO

Der grosse Vorteil des RSA-Verfahrens besteht darin, dass keine geheime Schlüsselinformation zwische dem Sender und dem Empfänger ausgetauscht werden muss. Vielmehr generiert der Empfänger sowol den öffentlichen und den privaten Schlüssel und teilt nur den öffentlichen Schlüssel dem Sender mit, behä aber den privaten Schlüssel bei sich geheim. Der Sender kann nun seine Daten verschlüsseln, aber nu der Empfänger kann sie entschlüsseln [mehr...].

In der Praxis wählt man die Primzahlen p und q sehr gross (mehrere Hundert Stellen lang). Di Generierung des öffentlichen Schlüssels erfordert nur die Produktbildung m = p * q, was sehr einfach is Will ein Hacker aus dem öffentlichen Schlüssel den privaten Schlüssel herausfinden, so muss er umgekeh aus m die beiden geheimen Primfaktoren bestimmen. Das Faktorisieren einer langen Zahl ist aber bishe nur mit einem enormen Rechenaufwand möglich. Kryptosysteme nutzen also die Grenzen de Berechenbarkeit.

Grundsätzlich gibt es aber kein absolut sichereres Verschlüsselungsverfahren. Die Verschlüsselung g aber bereits dann als sicher, wenn die Zeit für die Entschlüsselung wesentlich länger dauert als die Zei während der die Information von Wichtigkeit ist.

AUFGABEN

1. Versuch den Geheimtext
   AV SFX EXSWAXSXAFDTWMFZSZXJFXSTF
   CTFFSTUXJSXJKLSMES TDUFXMFSCGEEXF
   YJXMXSEAV
   ETP
   zu entschlüsseln. Es handelt sich um eine Caesar-Verschlüsselung.

   Bemerkung: Eine Lösungsmöglichkeit besteht darin, davon auszugehen, dass der Buchstabe E i deutschen Texten weitaus am häufigsten vorkommt. Du kannst aber auch al Verschiebungsmöglichkeiten durchprobieren.

2. Orientiere dich auf dem Internet, was man unter dem Verschlüsselungsverfahren mit einer Skytal versteht und implementiere einen Encoder/Decoder nach diesem Prinzip.

3. Begründe, warum die Caesar-Verschlüsselung ein Spezialfall des Vigenère-Verfahrens ist.

4. Erzeuge mit zwei Primzahlen p und q (beide kleiner als 100) einen öffentlichen und privaten Schlüss gemäss dem RSA-Verfahren und verschlüssle/entschlüssle damit einen Text.