1 RSA

RSA ist eine asymmetrische Verschlüsselung, d.h. zum Ver- und Entschlüsseln werden verschiedene Schlüssel verwendet. Normalerweise werden RSA-Schlüssel mit der ssl- oder ssh-Bibliothek erzeugt und liegen dann in einem **PEM**-Dateiformat vor. Die Schlüssel selbst sind reine **Zahlen**, allerdings sehr grosser Länge (z.B. 1024 bit).
Man kann die Schlüssel aber auch als reine Zahlen in einer csv-Datei speichern. Die einzelnen Teile einer Schlüsseldatei (**privater Schlüssel! Muss geheim beleiben!!!!**) sind:

N Hauptmodul N, N=p*q; p und q sind zwei Primzahlen

e Verschlüsselungsexponent; hat praktisch immer den Wert 65537 (

$$2^{16} + 1$$

d Entschlüsselungsexponent; lässt sich aus p und q berechnen; p und q sind geheim

p eine sehr grosse Primzahl

q eine ähnlich grosse Primzahl

Der öffentliche Schlüssel ist einfach nur die Kombination aus N und e. Der private Schlüssel enthält somit immer auch den öffentlichen Schlüssel. Das ist für das Entschlüsseln nicht notwendig, wird aber immer so gehandhabt.
Beispiel für die CSV-Datei mit einem superkurzen Schlüssel von 16bit:

```
52961;65537;48473;251;211
```

Die Zahlen in der Datei haben die Reihenfolge:

```
N;e;d;p;q
```

2 rsa-Modul

Da die Rechenoperationen von RSA aufwändig sind, soll eine fertige Library (Sybren A. Stüvel, sybren@stuvel.eu) verwendet werden:

```
#terminal starten
python3 -m venv rsa
cd rsa
source bin/activate
pip install rsa
```

Die Dokumentation zur Crypto-Lib ist hier:

```
https://stuvel.eu/python-rsa-doc/reference.html
```

3 Aufgaben

3.1 entschlüsseln

Schreibe ein Python-Programm, das die Datei `priv_key_raw.csv` öffnet und daraus N, e, d, p, q extrahiert und damit ein Objekt der Klasse

```
rsa.PrivateKey
```

erzeugt.

Lies die Datei `chiffre.txt` in ein einen String ein. Die Datei `chiffre.txt` ist eine grosse Zahl im Hexadezimal-Format, gespeichert als txt-Datei. Um diesen Textstring wieder in eine Zahl umzuwandeln, muss man die Funktion

```
bytes.fromhex(crypto_string)
```

verwenden. Diese Zahl kann dann mit der Methode

```
rsa.decrypt(crypto: bytes, priv_key: rsa.key.PrivateKey)
```

entschlüsselt werden.

Der return-Wert von decrypt ist ein Byte-Array, das muss noch mit der Methode decode('utf8') in eine Zeichenkette umgewandelt werden:

```
message.decode('utf8')
```

3.2 verschlüsseln

Schreibe ein Programm, das eine kurze Textdatei verschlüsselt und in der Datei `crypt.txt` speichert.

Als Schlüssel soll N und e aus folgendem Schlüssel verwendet werden:

```
https://dt.wara.de/pdf/sae/programmierung/python/publicKeyCryptographie
```

Um aus dem Text ein Bytes-Objet für die Verschlüsselungsmethode zu generieren, muss der Text encodiert werden:

```
meinText.encode('utf8')
```

Die Bytes des Chiffre-Objekts sollen als Text im hex-format gespeichert werden:

```
str(crypto.hex())
```

Achtung! Die Textdatei muss kürzer sein, als die Länge des oben erzeugten Schlüssels.

3.3 Schlüssel generieren

Schreibe ein Programm, das eine Schlüsseldatei `priv_key_raw.csv` im csv-Format erzeugt. Die Schlüsseldatei soll N, e, d, p, q in dieser Reihenfolge enthalten. Die Länge des Schlüssels soll einstellbar sein.

Zusätzlich soll eine Schlüsseldatei `pub_key_raw.csv` mit dem öffentlichen Schlüssel N, e erzeugt werden.

Dazu mit der Library ein key-Objekt erzeugen. Das key-Objekt enthält die Variablen N, e, d, p, q . Der Entwickler hat genau diese Bezeichner gewählt.