GSS Abgabe 05

Carolin Konietzny, Paul Bienkowski, Julian Tobergte, Oliver Sengpiel

16. Juni 2015

1.1. Ein symmetrisches Verschlüsselungssystem besitzt genau einen Schlüssel, mit dem zwischen Klartext und Chiffretext umgewandelt werden. Im Gegensatz dazu benötigt man für asymmetrische Verschlüsselung zwei verschiedene Schlüssel, je einen zum Ver- und Entschlüsseln der Nachricht.

1.2. a)
- Wenn Alice große Datenmengen verschicken möchte, oder zeitkritische Echtzeitanwendungen verschlüsseln möchte. Grund hierfür ist der i.d.R. höhere Rechenaufwand asymmetrischer Verschlüsselungssysteme.
- Wenn Alice auch Charlie, Dave und Erin dieselbe Nachricht zukommen lassen möchte, ohne den Inhalte mehrfach versenden zu müssen.

b) Alice erzeugt mit einem anderen, symmetrischen Verschlüsselungssystem einen neuen Schlüssel. Damit verschlüsselt sie die Daten. Den symmetrischen (Daten-)Schlüssel verschlüsselt sie ebenfalls, mit Bobs öffentlichen Schlüssel. Dann schickt sie beide Chiffres an Bob. Dieser kann den symmetrischen Schlüssel „entpacken" und damit die Nachricht dekodieren.

c) Wie genau die übertragene Nachricht aussieht, ist natürlich vom Verschlüsselungssystem abhängig. Auf jeden Fall beinhaltet sie zwei von jedem (durch das Protokoll definiert) zu unterscheidende Teile, einen für die Übertragung des symmetrischen Schlüssels (*header*), einen für den codierten Inhalt der Nachricht (*payload*).

2.2.

2.3.

3.2.

3.3.

5.2. Wir erhalten mit $d = 3243$ folgenden Klartext:

> Fuer die GSS-Klausur sind folgende Themen wichtig: Schutzziele, Angreifermodelle, Rainbow Tables, die (Un-)Sicherheit von Passwoertern und dazugehoerige Angriffe, Zugangs- und Zugriffskontrolle, Biometrische Verfahren, Timing-Attack und Power-Analysis, Grundlagen der Kryptographie, Authentifikationsprotokolle, das RSA-Verfahren und natuerlich alle anderen Inhalte, die wir in der Uebung und der Vorlesung behandelt haben :-)

Hier unser **python3**-code:

```
#!/usr/bin/env python3
from itertools import count

p, q, e, data = 281, 389, 67, [103625, 71396] # ...
```

```
n = p * q
p1q1 = (p - 1) * (q - 1)

for d in itertools.count():
    if (e * d) % p1q1 == 1:
        break

def decode(c):
    return chr((c ** d) % n)

print('d = ' + str(d))
print("".join(map(decode, data)))
```