GSS Abgabe 05

Carolin Konietzny, Paul Bienkowski, Julian Tobergte, Oliver Sengpiel

17. 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. Dieses Verfahren verschlüsselt immer noch nicht den tatsächlichen Datenverkehr, das heißt ein passiver Angreifer auf der Transportstrecke kann ungehindert mitlesen. Ebenfalls kann der Angreifer die gleiche Kombination aus r und c senden, um sich somit beim Server zu authentifizieren.

Verhindert wird lediglich ein Angriff, bei dem der Angreifer Benutzernamen und Passwort des Clients erfahren muss, etwa zur Verwendung in einem geschützten Login-System. Allerdings kann der Angreifer eine gefälschte Anfrage aus eigener Software senden, um dem Server vorzugaukeln, der Benutzer zu sein.

Ebenfalls kann der Angreifer die verschlüsselten Anmeldedaten nicht auf andere Systeme, die u.U. mit den gleichen Zugangsdaten gesichert wurden, zugreifen.

3.3. In diesem Verfahren kann ein Angreifer sich nicht ohne weiteres als der Benutzer ausgeben, denn hierfür ist die korrekte Response vonnöten. Da diese bei jeder Dienstanfrage unterschiedlich und vom Server ausgewählt wird, ist Kenntnis der Verschlüsselungsfunktion und des Schlüssels nötig.

Gelingt es dem Angreifer jedoch, die Kommunikation zwischen Benutzer und Dienstanbieter zu unterbrechen, und mit beiden gegenseitig in Echtzeit zu kommunizieren, kann der Angreifer die vom Dienstanbieter angeforderte Challenge abfangen, den Benutzer nach der dazugehörigen

Response fragen, und diese dem Server mitteilen, als hätte er sie selbst generiert. Dafür ist es jedoch nötig, dass der Angreifer dem Benutzer vorgaukelt, der Dienstanbieter zu sein, ein klassischer Man-in-the-Middle-Angriff.

Wie gehabt kann der Angreifer die folgende unverschlüsselte Kommunikation abhören und ggf. modifizieren, sofern ein Eingriff in die Netzwerkarchitektur den Angreifer als Teil der Transportstrecke agieren lässt.

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)))
```