## Aufgabe 1: Zentrale Begriffe der Kryptographie

(1) **Unterschiedliche Chiffren**: Bei einem **symmetrischen Kryptosystem** wird ein einziger Schlüssel zum ver- sowie entschlüsseln von Daten von beiden Kommunikationspartnern genutzt. Das zugrundeliegende Konzept von **asymmetrischen Kryptosystemen** sieht für jeden Kommunikationspartner hingegen jeweils einen öffentlichen und einen privaten Schlüssel vor. Die öffentlichen Schlüssel werden von den Kommunikationspartner ausgetauscht, steht einem der öffentliche Schlüssel eines Kommunikationspartners parat, so kann man Daten für diese Person verschlüsseln, die nur mit dem dazugehörigen privaten Schlüssel dieser Person wieder zu entschlüsseln sind.

(2a) Da symmetrische Verschlüsselungsverfahren performanter sind, wäre ein mögliches Einsatzszenario z.B. der Versand sehr großer Nachrichten / Datenmengen von Alice an Bob. Für sehr vertrauliche Daten eignet sich hingegen eher die Kommunikation mit asymmetrischen Verschlüsselungsverfahren, da hier eine höhere Sicherheit vorherrscht. Abgesehen davon ist die zu verwaltende Schlüsselmenge bei mehreren Kommunikationspartnern bei asymmetrischen Systemen kleiner.
Deshalb bietet sich allgemein ein hybrides Verfahren an, da alle oben aufgezählten Vorteile von symmetrischen und asymmetrischen Verfahren kombiniert werden: Sicherheit, Geschwindigkeit auch bei großen Datenmengen und einfacher zu verwaltende Schlüssel.

(2b) Konkret würde Alice dann die Nachricht $N$ mit einem zufällig generiertem Schlüssel $S_s$ symmetrisch verschlüsseln. Dieser symmetrische Schlüssel $S_s$ wird von Alice dann mit dem öffentlichen Schlüssel $K_p^B$ von Bob asymmetrisch verschlüsselt. Danach schickt Alice die Daten an Bob. Bob kann dann mit seinem privaten Schlüssel $K_s^B$ den verschlüsselten Schlüssel $S_s$ entschlüsseln und mit diesem dann die symmetrisch verschlüsselte Nachricht $N$ wieder entschlüsseln.

(2c) In der Nachricht $N$ sind die Daten, der eigentliche Kommunikationsinhalt gespeichert. Die Nachricht $N$ ist mit dem zufällig generierten Schlüssel $S_s$ symmetrisch verschlüsselt (z.B. mit AES, der Advanced Encryption Standard ist weitverbreitet und wird häufig genutzt). Der Schlüssel $S_s$ selbst wiederum ist mit dem öffentlichen Schlüssel $K_p^B$ von Bob asymmetrisch verschlüsselt. Sowohl $N$ als auch $S_s$ könnte Alice so sicher zusammen, z.B. in einer E-Mail an Bob, schicken. $N$ und $S_s$ könnten aber auch getrennt voneinander übermittelt werden.

## Aufgabe 2: Parkhaus

(2) **Schwächen des Systems**: Auf den Tickets ist zu erkennen, dass der Barcode an der ersten Stelle stets der gleiche zu sein scheint. So haben alle Besucher des Kinos den Barcode „32“ erhalten, der ihnen das Parken für 2,50 Eur ermöglicht. Außerdem tauchen noch Barcodes mit den Bezeichnungen „34“ und „36“ auf, zu denen vermutlich auch der Einzelhändler mit den vergünstigten Parktickets gehört. So könnten sich versierte Individuen also selbst diesen Barcode auf ihre Parktickets drucken um vergünstigt zu parken.

**Angreifermodell**:

a) **Rollen des Angreifers:** Außenstehender, Benutzer

b) **Verbreitung des Angreifers:** Der Angreifer hat lediglich Zugriff auf die Parktickets, nicht die Automaten

c) **Verhalten des Angreifers:** aktiv; vorhandene Parktickets werden verändert

d) **Rechenkapazität des Angreifers:** zum fälschen eines Parktickets wird keine besondere Rechenkapazität benötigt, der entsprechende Code muss nur korrekt aufgedruckt werden

(3) Anstatt den Rabattcode unverschlüsselt aufzudrücken, könnte man ein hybrides Verschlüsselungssystem nutzen. Man definiert fixe Rabattcodes, z.B. „R01“ für 2,50 Eur Rabatt. Das macht man für jeden Rabattvorteil den ein Geschäft vergeben können soll. Außerdem wählt man ein asymmetrisches Verfahren aus und erzeugt für den Kassenautomaten ein Schlüsselpaar, bestehend aus privatem und öffentlichen Schlüssel. Der öffentliche Schlüssel wird an die Geschäfte verteilt.
Bei Vergabe eines Rabatts beispielsweise vom Kino wird dann ein zufälliger Schlüssel $S$ für eine symmetrische Verschlüsselung erzeugt, mit dem man den Rabattcode, z.B. „R01“, verschlüsselt. Der Schlüssel $S$ wird dann mit dem öffentlichen Schlüssel des Kassenautomaten asymmetrisch verschlüsselt. Beides wird auf das Parkticket aufgedruckt. Der Kassenautomat entschlüsselt nun mit seinem privaten Schlüssel den Schlüssel $S$ und entschlüsselt danach den Rabattcode mit dem Schlüssel $S$.
Bei diesem System würde sich der aufgedruckte Rabattcode ständig ändern, weil er immer mit einem zufälligem symmetrischen Schlüssel verschlüsselt ist. Kein Angreifer könnte einen Rabattcode einfach kopieren. Da der private Schlüssel des Kassenautomaten geheim ist, könnte ein potentieller Angreifer auch nicht die geheimen Rabattcodes auslesen, da er den Schlüssel $S$ nicht entschlüsseln kann. Somit könnte er auch nicht einfach selbst die Rabattcodes erzeugen.

## Aufgabe 3: Authentifizierungsprotokolle

(2) Der Vorschlag des Systemadministrators zur Verbesserung des Systems macht keinen Unterschied. Solange die zufallsgenerierte Zahl nur vom Client generiert und vom Server nicht genutzt wird, könnte sowohl ein passiver als auch ein aktiver Angreifer den übermittelten, verschlüsselten String mitschneiden und zu einem späteren Zeitpunkt mittels Authentifikation an den Server senden. Die Zufallszahl müsste z.B. schon auf der Serverseite gespeichert werden und ein mehrmaliges Nutzen der gleichen Zufallszahl mit den gleichen Benutzerdaten verhindert werden, um diese Sicherheitslücke auszuschließen. Dieses Verfahren wäre aber nicht zielführend, da zum einen damit die möglichen, erfolgreichen Anmeldungen endlich wären und zum anderen im Verlaufe der Nutzung viel Overhead erzeugt werden würde, da mit der Zeit mit immer zahlreicher vorhandenen Zufallszahlen immer öfter eine erfolgreiche Anmeldung verweigert werden würde. Abgesehen davon verbraucht dieses Verfahren mit der Zeit auch immer mehr Speicherplatz.

(3) Das Mitschneiden von einem erfolgreichen Login kann nicht mehr genutzt werden, um einfach den gleichen übermittelten, verschlüsselten String dem Dienstanbieter zu senden, denn der Dienstanbieter gibt hier eine spezifische, selbsterzeugte Zufallszahl vor. Da es sich um ein einseitiges Challenge-Response-Verfahren handelt, könnte sich ein potentieller Angreifer immernoch als Dienstanbieter ausgeben/tarnen, um so die Benutzerdaten eines Nutzers zu erbeuten. Ein Bruteforceangriff wäre auch denkbar.

## Aufgabe 4: „Mensch ärgere Dich nicht“ über das Telefon

## Aufgabe 5: RSA-Verfahren

(2) In der Aufgabe waren die Werte von $p, q, e$ gegeben mit: $p = 281, q = 389, e = 67$

Mithilfe von Wikipedia fanden wir heraus, dass zur Entschlüsselung zwei wichtige Komponenten berechnet werden müssen, $N$ und $d$.
Bei RSA ist $N$ durch $N = p * q$ gegeben. $d$, der Entschlüsselungsexponent, wiederum ist das multiplikative inverse von $e$ modulo $((p-1)*(q-1))$.

Für $N$ ergibt sich also: $109.309 = p * q = N$.

Um $d$ zu berechnen verwendeten wir Wolframalpha:

```
(281-1) * (389-1)                 # evaluiert zu 108.640
inverse of 67 modulo 108640       # evaluiert zu   3.243
```

Damit hatten wir alle wichtigen Parameter für unser Programm:
$p = 281,\ q = 389,\ e = 67,\ d = 3243,\ N = 109.309$

Unser Programm sieht dabei wie folgt aus (Python):

```
1   # Eine Liste mit allen verschlüsselten Werten
2   secretdata = [103625, ..., 62098]
3
4   p = 281       # gegeben in der Aufgabenstellung
5   q = 389       # gegeben in der Aufgabenstellung
6   e = 67        # gegeben in der Aufgabenstellung
7   N = 109309    # berechnet
8   d = 3243      # berechnet
9
10  # Erzeugt eine Liste, indem jeder Wert von secretdata
11  # aufgerufen und mit der lambda-Funktion bearbeitet wird:
12  # x ist dabei ein einzelner Wert der Liste secretdata.
13  # str() wandelt die Rückgabe von uncichr() in einen
14  # String um und unichr() gibt einen Unicode für ein
15  # bestimmtes Zeichen zurück. Die Berechnung x**d % N
16  # berechnet das Ergebnis von x hoch d modulo N.
17  # Diese Formel haben wir ebenfalls Wikipedia entnommen.
18  # Jeder verschlüsselte Wert x wird so also entschlüsselt
19  # und dann in ein lesbares Zeichen umgewandelt.
20  decryptData = map(lambda x: str(unichr(x**d % N)), secretdata)
21
22  # Erzeugt aus der Liste decryptData, die die
23  # entschlüsselten Werte enthält, einen einzigen String
24  decryptStr = "".join(decryptData)
25
26  print decryptStr    # Ausgabe auf der Konsole
```

Die einzige Zeile, die den verschlüsselten Text entschlüsselt und tatsächlich etwas berechnet ist also in unserem Programm Zeile 20. Alle anderen Zeilen dienen lediglich als Datencontainer, der besseren Lesbarkeit oder Konsolenausgabe.

Der entschlüsselte Text lautet:

> 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 :-)