Modul 63211 – Verteilte Systeme
Einsendeaufgaben 4 – WS 2025/2026
FernUniversität Hagen, Lehrgebiet Kooperative Systeme

Aufgabe 4.1: Diffie-Hellman (15 + 15) + (4 + 4 + 4 + 4 + 4) Punkte

1. Entwickeln Sie ein zweiteiliges Python-Programm, bei dem Client und Server über TCP einen gemeinsamen *Sitzungsschlüssel* (*session key*) mit dem *Diffie-Hellman-Verfahren* herstellen, siehe im Buch Seite 565 und dort auch Figure 9.5 (Distributed Systems, 4th edition, Version 4.02X, 2024).

   Für die Berechnung von $g^x \bmod n$ sollen Sie zur Vermeidung allzu großer Zwischenergebnisse den folgenden *square-and-multiply-Algorithmus* verwenden.

   Hier wird $x = \sum_{i=0}^{l-1} x_i \cdot 2^i,\ x_i \in \{0,1\}$ als Summe von Zweierpotenzen dargestellt, oder anders gesagt, die x_i sind die Bits in der Binärdarstellung der Zahl x mit l Stellen.

   Square-and-Multiply-Algorithmus

   Eingabe: g, $x = \sum_{i=0}^{l-1} x_i \cdot 2^i$, n

   1. $z := 1$
   2. **for** $i := l-1$ **downto 0 do**
   3. $\quad z := z^2 \bmod n$
   4. $\quad$ **if** $x_i = 1$ **then** $z := (z \cdot g) \bmod n$

   Ausgabe: $z = g^x \bmod n$

   Um die x_i zu bestimmen, können Sie die folgende Methode verwenden:

   Eingabe: $x > 0$ ganzzahlig
   1. $i := 0$, $s := x$
   2. **while** $s > 0$ **do**
   3. $\quad x_i := s \bmod 2$ /* 0 oder 1 bei gerade oder ungerade */
   4. $\quad s := (s - x_i) \div 2$ /* ganzzahlige Division ohne Rest */
   5. $\quad i := i + 1$

   Ausgabe: $l = i - 1$, Liste $x_0, \ldots, x_{l-1}$

2. Welche gemeinsamen Schlüssel von Client und Server ergeben sich für die folgenden Fälle?

   a) $n = 27803$ $g = 5$ $x = 21131$ $y = 17555$
   b) $n = 27803$ $g = 5$ $x = 169$ $y = 23456$
   c) $n = 27001$ $g = 101$ $x = 21768$ $y = 9898$
   d) $n = 30113$ $g = 52$ $x = 8642$ $y = 24673$
   e) $n = 30113$ $g = 52$ $x = 28654$ $y = 12385$

   Diese Zahlen sind nur zu Testzwecken recht klein und so natürlich nicht sicher gegen Brute-Force-Angriffe.

Aufgabe 4.2: man-in-the-middle 15 Punkte

Alice will einen sicheren Kanal zu Bob mit dem öffentlichen Schlüssel von Bob aufbauen. Die folgenden Schritte erfolgen:

1. Alice fragt Bob nach seinem öffentlichen Schlüssel K_B^+. Bob schickt ihn an Alice.
2. Alice erzeugt einen geheimen gemeinsamen Schlüssel K_{AB} und verschlüsselt ihn mit K_B^+, sie sendet diese Nachricht an Bob.
3. Bob entschlüsselt die Nachricht und erhält den Schlüssel K_{AB}.

Wie könnte hier ein *man-in-the-middle-Angriff* stattfinden?

Aufgabe 4.3: Sicherer Verbindungsaufbau 35 Punkte

Zur Vermeidung eines man-in-the-middle-Angriffs speichert beim *Needham-Schroeder-Protokoll* ein *key distribution center* (KDC) alle angemeldeten Hosts und die mit ihnen jeweils vereinbarten und geteilten geheimen Schlüssel, z. B. $K_{A,KDC}$ für Host A usw., siehe Note 9.5 (Advanced: The Needham-Schroeder protocol, Seite 577 im Buch Distributed Systems, 4th edition, Version 4.02X, 2024). Wenn nun Host A mit Host B kommunizieren möchte, generiert das KDC dafür einen geheimen Schlüssel $K_{A,B}$.

Bei der Kommunikation per öffentlichen Schlüsseln dagegen, siehe Seite 569 im Buch, werden keine geteilten geheimen Schlüssel benötigt, nur muss jeder Host den öffentlichen Schlüssel von jedem anderen Host kennen.

Wir wollen nun diese beiden Ideen verbinden zu einem System, bei der ein KDC die öffentlichen Schlüssel aller Hosts kennt und bei Bedarf diese herausgibt. Entwerfen Sie also ein solches KDC-System angelehnt an das Needham-Schroeder-Protokoll, bei dem nur öffentliche Schlüssel verwendet werden und zwei Hosts mit Hilfe des KDC eine sichere Verbindung miteinander aufbauen können.

Ausgangszustand ist eine Menge von Hosts und das KDC, alle haben ein eigenes Schlüsselpaar generiert. Alle Hosts sind beim KDC registriert, so dass dort ihre öffentlichen Schlüssel vorliegen, während nur sie selbst jeweils ihre privaten Schlüssel kennen. Der öffentliche Schlüssel des KDC ist überall bekannt.

Welche Schritte benötigt nun das *Authentifikationsprotokoll*, wenn Host A mit Host B einen abgesicherten Kommunikationskanal eröffnen will?