Verteilte Systeme – Übungsblatt 7: Sicherheit

Prof. Dr. Oliver Haase, Dr. Thomas Zink

In dieser Übung sollen Sie Aqualife um Sicherheitsmechanismen erweitern. Die dazu notwendigen Code-Änderungen sollen in einer von Ihnen zu erstellenden Subklasse `SecureEndpoint` der bisher verwendeten Klasse `Endpoint` gekapselt werden, so dass der Aqualife-Code bis auf die Erzeugung der neuen Endpunkte nicht geändert werden muss.

In dieser Übung müssen sie einige Java-Plattform-Klassen zur Schlüsselerzeugung und Verschlüsselung verwenden. Konsultieren Sie deshalb bitte auch die entsprechende Java-API-Dokumentation.

Aufgabe 1 – symmetrische Verschlüsselung

Zunächst ist jede Kommunikation zwischen Aqualife-Clients und zwischen Clients und dem Broker mit Hilfe symmetrischer Schlüssel zu verschlüsseln. Um für diese Aufgabe den Schlüsselaustausch zu vermeiden, sollen alle Clients sowie der Broker denselben gemeinsamen Schlüssel verwenden. Gehen Sie zur Lösung der Aufgabe konkret wie folgt vor:

- Die von Ihnen zu erstellende Klasse `SecureEndpoint` muss eine Subklasse von `Endpoint` sein, um diese ersetzen zu können. Intern soll `SecureEndpoint` einen eingekapselten, herkömmlichen `Endpoint` für die eigentliche Kommunikation verwenden.

- Der `SecureEndpoint` muss den symmetrischen Schlüssel erzeugen und verwalten (speichern). Nutzen Sie zum Erzeugen des Schlüssels die Plattform-Klasse `SecretKeySpec`. Verwenden Sie als "'Key-Material"' für den Konstruktor die Bytefolge, die Sie aus dem String "CAFEBABECAFEBABE" erzeugen. Damit ist sichergestellt, dass alle Endpunkte denselben Schlüssel erzeugen. Als Verschlüsselungsalgorithmus geben Sie `"AES"` an.

- Der `SecureEndpoint` benötigt außerdem zwei `Cipher`-Objekte, eines zum Verschlüsseln und eines zum Entschlüsseln. Verwenden Sie zum Erzeugen der Objekte die Fabrikmethode `Cipher.getInstance(String transformation)`, und geben

Sie als Transformation wiederum den Verschlüsselungsalgorithmus `"AES"` an. Anschließend muss eines der Objekte zum Verschlüsseln und eines zum Entschlüsseln initialisiert werden.

- Die `send`-Methode der Klasse `SecureEndpoint` soll nun zunächst den Payload-Teil der zu versendenden Nachricht mit Hilfe des Verschlüsseler verschlüsseln und danach die verschlüsselte Nachricht über den internen normalen `Endpoint` an den gewünschten Empfänger verschicken.
- Analog sollen die beiden `receive`-Methoden ankommende Nachrichten von ihrem internen Endpunkt entgegennehmen, die Payloads mit Hilfe des entsprechenden `Cipher`s entschlüsseln und anschließend an den Aufrufer zurückgeben.
- Ersetzen Sie schließlich im Broker und im Client den normalen `Endpoint` durch den neuen `SecureEndpoint`, so dass alle Kommunikation verschlüsselt abläuft.

Aufgabe 2 – asymmetrische Verschlüsselung und Schlüsseltausch

Verwenden Sie in dieser Aufgabe statt eines gemeinsamen symmetrischen Schlüssels für alle Clients (und den Broker) individuelle asymmetrische Schlüsselpaare. Dazu müssen zwei Kommunikationspartner, die zum ersten Mal miteinander kommunizieren, vor dem eigentlichen Nachrichtenaustausch zuerst ihre öffentlichen Schlüssel austauschen. In einem realistischen Szenario würde dies mittels Zertifikaten geschehen; in dieser Aufgabe werden einfach die unsignierten öffentlichen Schlüssel versendet. Gehen Sie zur Lösung dieser Aufgabe wie folgt vor:

- Statt eines symmetrischen Schlüssels erzeugt die Klasse `SecureEndpoint` nun ein asymmetrisches Schlüsselpaar mit Hilfe der Java-Klasse `KeyPairGenerator`. Verwenden Sie als Algorithmus `"RSA"`.
- Die Klasse `SecureEndpoint` benötigt nun eine Datenstruktur, in der sie sich für jeden Kommunikationspartner, mit dem bereits kommuniziert wurde, dessen öffentlichen Schlüssel merkt.
- Wenn in Folge eines `send`-Aufrufs eine Nachricht versandt werden soll, wird erst überprüft, ob der öffentliche Schlüssel des Empfängers bereits bekannt ist. Falls ja, wird die Nachricht entsprechend verschlüsselt. Falls nein, muss zuerst ein Schlüsselaustausch stattfinden. Definieren Sie dazu einen neuen Nachrichtentyp `KeyExchangeMessage`. Beachten Sie, dass diese Nachrichten auf der empfangenden Seite nicht nach oben gelangen dürfen, sondern nur zur Kommunikation zwischen zwei `SecureEndpoint`s verwendet werden dürfen.
- In den `receive`-Methoden müssen Sie zum einen prüfen, ob es sich bei der empfangenen Nachricht um eine `KeyExchange`-Nachricht handelt und zum anderen alle anderen Nachrichten korrekt entschlüsseln, bevor sie Sie nach oben reichen.

- Verwenden Sie zum Ver- und Entschlüsseln, ähnlich wie in Aufgabe 1, geeignete Cipher-Objekte.

Viel Spass & gutes Gelingen!