Sicherheit in verteilten Systemen
Praktische Anwendung

Andreas Stadelmeier

Duale Hochschule Baden-Württemberg, Campus Horb

1 Vorstellung der Anwendung

1.1 Überblick

Die hier vorgestellte Webanwendung ist ein Chatprogramm zur Kommunikation zweier Gesprächspartner mittels Kurznachrichten. Über einen Webbrowser kann die Anwendung von einem zentralen Server bezogen werden. Nach einer Authentifizierung am Server ist der Client in der Lage Nachrichten zu senden und zu empfangen. Dabei ist die Besonderheit der Webanwendung, dass Nachrichten nicht als Klartext zum Server gesendet, sondern schon auf der Clientseite verschlüsselt werden. Der Server dient nur als Vermittler zwischen den einzelnen Kommunikationspartnern. Bei einem Chat-System ohne Clientseitige Verschlüsselung werden Daten nur zur Übertragung zwischen zwei Knotenpunkten verschlüsselt übertragen. Nach der Übertragung liegen sie auf der Gegenseite wieder im Klartext vor und könnten von dieser gelesen oder manipuliert werden. Bei der vorgestellten Anwendung dagegen ist der Server nicht in der Lage die verschlüsselten Nachrichten zu lesen. Dadurch kann auch ein Angreifer, welcher an die vom Server gespeicherten Daten gelangt, die geführten Kommunikationen nicht entschlüsseln. Außerdem schützt die Verschlüsselung die privaten Daten der Gesprächspartner. Kein Administrator oder automatischer Scanvorgang ist in der Lage die übermittelten Nachrichten zu dechiffrieren.

2 Frameworks und Code-Bibliotheken

Dieses Kapitel stellt die Frameworks und Bibliotheken vor, welche dazu verwendet wurden die Anwendung umzusetzen.

2.1 NodeJs

Node.js ist eine Plattform mit der sich schnelle Netzwerkanwendungen entwickeln lassen. Sie basiert auf der V8 JavaScript Engine, einen Open source Javascript-Interpreter von Google. Node.js erweitert diesen Interpreter um eine Programmierschnittstelle, welche speziell für die Entwicklung eines Verteilten Systems ausgelegt ist. Node.js wird in diesem Projekt für den Aufbau der Server-Logik verwendet. Quelle: http://nodejs.org/

2.2 NowJs

NowJs ist ein Framework für NodeJs. Es unterstützt den Aufbau einer Echtzeit Webanwendung. NodeJs wird dabei auf Serverseite in Form eines NodeJs-Plugins und auf Clientseite als Javascript-Bibliothek eingesetzt. NowJs übernimmt den Aufbau einer bidirektionalen Verbindung zwischen der Webanwendung auf Clientseite und der Serveranwendung. Zudem bietet das Framework eine Sitzungsverwaltung, welche es ermöglicht, dass sich ein Client pro Sitzung nur einmalig am Server authentifizieren muss.

2.3 Stanford Javascript Crypto Library

Die Stanford Javascript Crypto Library (SJCL) ist eine Javascript-Bibliothek, welche Funktionen zur Verschlüsselung von Daten bzw. Text anbietet. Sie wird von der Webanwendung auf Clientseite eingesetzt um Daten mittels einer synchronen Verschlüsselung zu ver- und entschlüsseln.

2.4 RSA-Crypto Library

Diese Javascript-Bibliothek wird eingesetzt um RSA-Schlüsselpaare zu generieren und um Nachrichten mithilfe der Schlüsselpaare zu ver- und entschlüsseln.

3 Sicherheitsmechanismen der Anwendung

Im Folgenden werden die implementierten Sicherheitsmechanismen der Anwendung vorgestellt.

RSA Die Asynchrone Verschlüsselungstechnik ist der zentrale Kern der Kommunikation. Jeder Benutzer erzeugt bei der Erstellung eines Accounts einen Privaten und Öffentlichen Schlüssel. Der öffentliche Schlüssel wird auf den Server geladen und dadurch öffentlich gemacht. Andere Benutzer können Nachrichten vor dem Absenden mit dem öffentlichen Schlüssel des Empfängers verschlüsseln. Anschließend kann sie nur der Benutzeraccount mit dem passenden privaten Schlüssel dechiffrieren.

AES Die Synchrone Verschlüsselung wird benutzt um den privaten Schlüssel eines RSA-Schlüsselpaars zusätzlich durch ein Passwort zu schützen. Der Algorithmus chiffriert den Schlüssel mit einem Passwort, das nur der Besitzer kennt.

HTTPS Das Hypertext Transfer Protocol Secure wird als Protokoll für die Kommunikation zwischen den einzelnen Clients und dem Server verwendet. Das Protokoll baut zum einen eine abhörsichere Verbindung auf und verhindert zusätzlich Man-in-the-middle-Angriffe. Dieser Schutz ist notwendig, da bei jedem Start der Webanwendung diese vom Server zum Client übertragen wird und für einen Man-in-the-middle-Angriff bei diesem Schritt die Möglichkeit besteht Schadcode einzuschleusen.

4 Implementierung

4.1 Account erstellen

Zur Erstellung eines Accounts muss der Benutzer einen Benutzernamen und ein Passwort wählen. Während dessen erstellt die Webanwendung mithilfe der RSA-Bibliothek (siehe Kapitel 2.4) ein RSA-Schlüsselpaar. Sobald dieses generiert wurde und ein Benutzername sowie ein Passwort feststeht wird der private Schlüsselteil mit dem Benutzerpasswort verschlüsselt. Dadurch entsteht ein

zusätzlicher Schutz, welcher bei einem Diebstahl des Schlüssels vor einer Dechiffrierung sämtlicher Nachrichten des Besitzers schützt. Anschließend werden Benutzername, Passwort und das Schlüsselpaar an den Server übermittelt. Dieser speichert die übermittelten Daten in einer Datenbank, wobei das Passwort nur als Hashwert abgelegt wird. Den Ablauf des Registrierungsvorgangs auf Clientseite zeigt das Sequenzdiagramm in Abbildung 1.

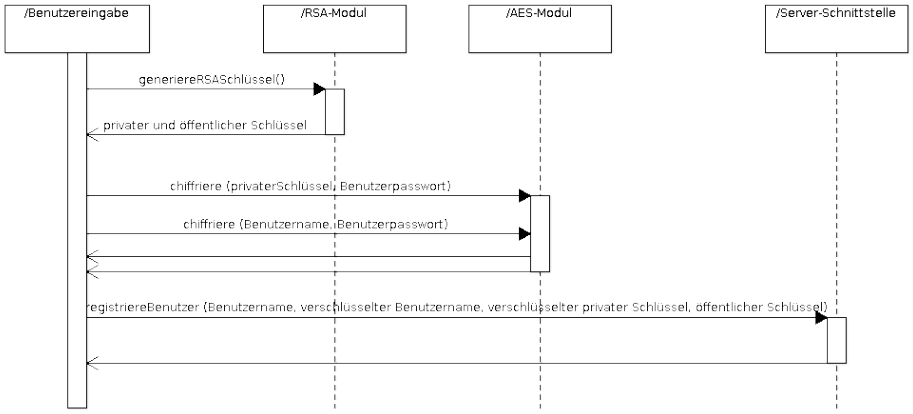

Abbildung 1. Ablauf der Generierung eines Benutzeraccounts auf Clientseite

4.2 Authentifizierung am Server

Bevor ein Client Nachrichten an andere Benutzer versenden darf muss er sich am Server authentifizieren. Jeder am Server registrierte Benutzer (siehe Kapitel 4.1) besitzt einen Account mit einem eindeutigen Benutzernamen und einem Passwort. Mit diesen Informationen authentifiziert er sich am Server. Dieser weist ihm eine Sitzung zu und übermittelt dessen privaten Schlüssel. Der private Schlüsselteil muss anschließend noch mittels des Benutzerpassworts entschlüsselt werden.

4.3 Nachricht Versenden

Das Versenden einer Nachricht findet auf Clientseite statt. Die komplette Programmlogik ist in Javascript umgesetzt und kann von aktuellen Internet-Browsern ausgeführt werden. In Abbildung 2 zeigt ein Sequenzdiagramm den Ablauf dieses Vorgangs.

Abbildung 2. Sequenzdiagramm für das Versenden einer Nachricht

4.4 Nachrichten empfangen

Zum Empfangen von Nachrichten muss die Webanwendung diese vom Server anfordern. Zuvor ist eine Anmeldung am Server notwendig. Dadurch ist dieser in der Lage, nur die Nachrichten zu übermitteln, welche für den angemeldeten Benutzer bestimmt sind.
Die empfangenen Daten sind mit dem öffentlichen Schlüssels des Empfängers verschlüsselt. Die Anwendung muss sie zuerst mit dem privaten Schlüssel des Benutzers entschlüsseln. Anschließend können die Nachrichten im Klartext dargestellt werden. Der Ablauf ist als Sequenzdiagramm in Abbildung 3 dargestellt.

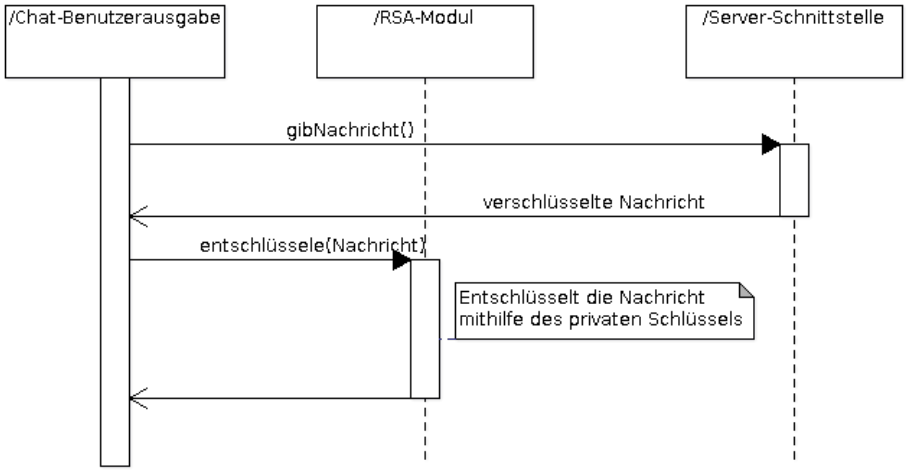

Abbildung 3. Sequenzdiagramm für das Empfangen einer Nachricht

4.5 Server

Der Server ist für das Zwischenspeichern von Textnachrichten, der Kommunikation mit den einzelnen Clients, das Speichern von Benutzeraccounts und das Bereitstellen der Webanwendung verantwortlich. Außerdem enthält er Programmlogik zur Authentifizierung der Benutzer.
Seine Struktur ist in Abbildung 4 als Klassendiagramm dargestellt. Die Struktur teilt sich in zwei Bereiche auf. Zum einen einen die Chat-Logik, welche die Datenbank verwaltet und darin Nachrichten und Benutzeraccounts speichert, zum anderen die Schnittstelle zu den Clients. Die Kommunikation zwischen Server und Client erfolgt über das HTTP-Protokoll. Der HTTP-Request-Handler verarbeitet und beantwortet Anfragen der Clients. Er ist für die Übertragung der Webanwendung zum Benutzer verantwortlich. Den anschließenden Austausch von Informationen in Echtzeit übernimmt die Now-Klasse, eine Implementierung des NowJs-Frameworks (siehe Kapitel 2.2). Auf der Serverseite stellt der HTTP-Server die passende Schnittstelle bereit, welche die Klassen 'Now' und 'HTTP-Request-Handler' benutzen. Diese sind für den eigentlichen Datenaustausch zuständig.

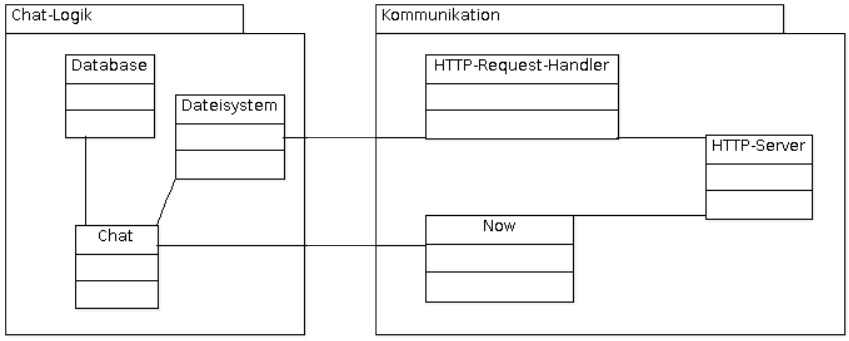

Abbildung 4. Struktur des Servers als Klassendiagramm