Implementación de Comunicaciones Seguras Cliente-Servidor

Programación de Servicios y Procesos

febrero de 2025

Índice

1. Introducción

La seguridad en las comunicaciones es un pilar fundamental en el desarrollo de aplicaciones modernas, especialmente en un entorno donde las amenazas y ataques cibernéticos son cada vez más frecuentes y sofisticados. Esta práctica busca ayudar a adquirir las habilidades necesarias para implementar comunicaciones seguras, utilizando técnicas de cifrado avanzadas y sistemas de autenticación robustos. A lo largo de esta práctica, se abordará el desarrollo de aplicaciones con comunicaciones seguras mediante el uso de cifrado simétrico y asimétrico, además de implementar un robusto sistema de autenticación basado en retos. Se proporcionan material teórico así como referencias adicionales para complementar el aprendizaje práctico, asegurando que se puedan aplicar estos conceptos críticos de seguridad en escenarios reales.

Resumen de la Práctica:

1. **Tarea 1: Cifrado Simétrico** - Implementación de cifrado AES para asegurar la comunicación entre un cliente y un servidor.
2. **Tarea 2: Cifrado Asimétrico** - Uso de RSA para cifrado y descifrado de mensajes, proporcionando una capa adicional de seguridad.
3. **Tarea 3: Autenticación Basada en Retos** - Desarrollo de un sistema de autenticación que utiliza retos criptográficos para verificar la identidad de los usuarios.

Estas tareas combinan teoría y práctica para implementar soluciones de seguridad efectivas en aplicaciones modernas.

2. Objetivo de la Práctica

- Comprender y aplicar técnicas de cifrado simétrico y asimétrico.
- Desarrollar una aplicación cliente-servidor con comunicaciones seguras.
- Implementar y evaluar un sistema de autenticación basado en retos.
- Mejorar la capacidad de documentación técnica y pruebas de seguridad.

3. Tareas a Desarrollar

3.1. Tarea 1: Configuración Inicial y Cifrado Simétrico

Descripción de cómo inicializar y configurar el entorno de desarrollo, junto con la implementación del cifrado simétrico.

3.2. Tarea 2: Cifrado Asimétrico

Proceso de implementación y prueba del cifrado asimétrico, asegurando la integridad y confidencialidad de la comunicación.

3.3. Tarea 3: Sistema de Autenticación Basado en Retos

Desarrollo de un sistema de autenticación que emplea retos criptográficos para confirmar la identidad de los usuarios antes de permitir el acceso a recursos críticos. Este sistema es esencial para garantizar que las comunicaciones entre el cliente y el servidor sean seguras y que solo los usuarios autorizados puedan acceder a la información sensible.

4. Tarea 2: Cifrado Asimétrico

Esta tarea se centra en la implementación del cifrado asimétrico, proporcionando una capa adicional de seguridad mediante el uso de claves públicas y privadas. El cifrado asimétrico es crucial para operaciones como la autenticación y la firma digital, además de la encriptación de datos.

- **Generación de Claves:** Configura un par de claves utilizando RSA, uno de los algoritmos de cifrado asimétrico más robustos y comúnmente usados.

```
1 KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("
     RSA");
2 keyPairGen.initialize(2048);
3 KeyPair keyPair = keyPairGen.generateKeyPair();
```

- **Cifrado y Descifrado de Mensajes:** Implementa funciones para cifrar mensajes utilizando la clave pública y descifrarlos con la clave privada.

```
1      Cipher cipher = Cipher.getInstance("RSA/ECB/
          PKCS1Padding");
2      cipher.init(Cipher.ENCRYPT_MODE, keyPair.getPublic());
3     byte[] encrypted = cipher.doFinal(message.getBytes());
```

4.1. Tarea 3: Sistema de Autenticación Basado en Retos

Desarrollo de un sistema de autenticación que utilice mecanismos de desafío-respuesta para verificar la identidad de los usuarios.

5. Tarea 3: Sistema de Autenticación Basado en Retos

Esta tarea se enfoca en el desarrollo e implementación de un sistema de autenticación basado en retos, utilizando el cifrado asimétrico para asegurar la integridad y confidencialidad del reto y de la respuesta.

- **Descripción del Reto:** El servidor genera un reto único que podría ser, por ejemplo, un nonce (número que se usa una sola vez) o un hash de algún dato significativo, y lo envía al cliente cifrado con la clave pública del cliente.

```
1         byte[] nonce = new byte[16];
2         new SecureRandom().nextBytes(nonce);
3         cipher.init(Cipher.ENCRYPT_MODE, clientPublicKey);
4         byte[] encryptedNonce = cipher.doFinal(nonce);
```

- **Respuesta del Cliente:** El cliente recibe el reto, lo descifra utilizando su clave privada, realiza alguna operación predefinida (como firmar el nonce o responder con algún dato relacionado), y envía la respuesta de vuelta al servidor cifrada con la clave pública del servidor.

```
1      cipher.init(Cipher.DECRYPT_MODE, privateKey);
2      byte[] decryptedNonce = cipher.doFinal(encryptedNonce);
3      // El cliente responde con una operacion sobre el nonce
4      cipher.init(Cipher.ENCRYPT_MODE, serverPublicKey);
5      byte[] encryptedResponse = cipher.doFinal(response);
```

- **Verificación en el Servidor:** El servidor descifra la respuesta, verifica la validez de la respuesta según el reto inicial, y autentica al cliente si la respuesta es correcta.

```
1 cipher.init(Cipher.DECRYPT_MODE, serverPrivateKey);
2 byte[] decryptedResponse = cipher.doFinal(encryptedResponse)
     ;
3 // Verificar la respuesta
4 if (Arrays.equals(decryptedResponse, expectedResponse)) {
5     System.out.println("Cliente autenticado con exito.");
6 } else {
7     System.out.println("Autenticacion fallida.");
8 }
```

5.1. Tarea 3: Sistema de Autenticación Basado en Retos

Esta tarea se centra en desarrollar un sistema de autenticación robusto que utilice retos criptográficos para verificar la identidad de los comunicantes. Este sistema es esencial para asegurar que tanto el cliente como el servidor son quienes dicen ser antes de permitir el acceso a información sensible o realizar transacciones críticas.

- **Generación y Envío de Retos:** El servidor generará un reto criptográfico, como un nonce o un hash de un dato relevante, y lo enviará al cliente. Este reto deberá ser cifrado utilizando la clave pública del cliente para asegurar su confidencialidad.

```
1 SecureRandom random = new SecureRandom();
2 byte[] challenge = new byte[16];
3 random.nextBytes(challenge);
4 Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
5 cipher.init(Cipher.ENCRYPT_MODE, clientPublicKey);
6 byte[] encryptedChallenge = cipher.doFinal(challenge);
```

- **Respuesta del Cliente:** El cliente, al recibir el reto, deberá descifrarlo utilizando su clave privada, responder adecuadamente según el protocolo definido (por ejemplo, firmar el reto y devolverlo), y cifrar la respuesta utilizando la clave pública del servidor antes de enviarla de vuelta.

```
cipher.init(Cipher.DECRYPT_MODE, clientPrivateKey);
byte[] decryptedChallenge = cipher.doFinal(
    encryptedChallenge);
// El cliente anade su firma o realiza la operaci\'on
    requerida
cipher.init(Cipher.ENCRYPT_MODE, serverPublicKey);
byte[] encryptedResponse = cipher.doFinal(responseBytes);
```

- **Verificación del Reto en el Servidor:** El servidor, al recibir la respuesta cifrada, la descifrará con su clave privada y verificará la validez de la respuesta para confirmar la identidad del cliente.

```
cipher.init(Cipher.DECRYPT_MODE, serverPrivateKey);
byte[] decryptedResponse = cipher.doFinal(encryptedResponse)
    ;
// Verificar la validez de la respuesta
if (Arrays.equals(decryptedResponse, expectedResponse)) {
    System.out.println("Autenticacion exitosa.");
} else {
    System.out.println("Error de autenticaci\'on.");
}
```

Documentación: Documenta todos los pasos y el código desarrollado. Comenta el código para explicar la funcionalidad de las secciones clave y métodos utilizados.

6. Criterios de Evaluación

La evaluación de la práctica se basará en los siguientes criterios, cada uno contribuyendo a la nota final según el porcentaje especificado:

- **Correcta Implementación del Cifrado** (30 %): Evaluación de la correcta aplicación de las técnicas de cifrado simétrico y asimétrico. Se verificará que los datos se cifren y descifren adecuadamente.
- **Funcionalidad del Sistema de Autenticación** (30 %): Evaluación de la efectividad del sistema de autenticación basado en retos. Se verificará que el sistema rechace accesos no autorizados y permita los autorizados de manera segura.
- **Calidad del Código** (20 %): Se evaluará que el código sea claro, bien organizado y debidamente comentado.
- **Documentación** (20 %): Evaluación de la calidad de la documentación proporcionada, incluyendo la explicación de cómo funciona el sistema y cómo se han implementado las soluciones.

Cada sección contribuirá hasta un máximo de puntos según el porcentaje asignado, con la nota total posible sumando hasta un máximo de 100 %. Los estudiantes deben esforzarse en cada área para maximizar su puntuación final.

7. Recursos Adicionales

Para facilitar el desarrollo de la práctica, los siguientes recursos pueden ser de utilidad:

- Documentación oficial de Java sobre cifrado: https://docs.oracle.com/javase/8/docs/api/javax/crypto/package-summary.html
- Tutoriales de RSA y AES: https://www.tutorialspoint.com/cryptography_with_python/index.htm
- Herramientas para pruebas de seguridad en aplicaciones web, tales como OWASP ZAP: https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project