JUST DO IT
HÖHERE ABTEILUNG FÜR
INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Laborprotokoll

DezSys05 - Java Security

Systemtechnik-Labor
5BHIT 2015/16, Gruppe Z

Michael Weinberger

Version 0.1
Begonnen am 15. Januar 2016
Beendet am ???

Note:
Betreuer: Micheler

Inhaltsverzeichnis

1 Einführung

Diese Übung zeigt die Anwendung von Verschlüsselung in Java.

1.1 Ziele

Das Ziel dieser Übung ist die symmetrische und asymmetrische Verschluesselung in Java umzusetzen. Dabei soll ein Service mit einem Client einen sicheren Kommunikationskanal aufbauen und im Anschluss verschlüsselte Nachrichten austauschen. Ebenso soll die Verwendung eines Namensdienstes zum Speichern von Informationen (hier PublicKey) verwendet werden.
Die Kommunikation zwischen Client und Service soll mit Hilfe einer Übertragungsmethode (IPC, RPC, Java RMI, JMS, etc) aus dem letzten umgesetzt werden.

1.2 Voraussetzungen

- Grundlagen Verzeichnisdienst
- Administration eines LDAP Dienstes
- Grundlagen der JNDI API für eine JAVA Implementierung
- Grundlagen Verschluesselung (symmetrisch, asymmetrisch)
- Einfuehrung in Java Security JCA (Cipher, KeyPairGenerator, KeyFactory)
- Kommunikation in Java (IPC, RPC, Java RMI, JMS)
- Verwendung einer virtuellen Instanz für den Betrieb des Verzeichnisdienstes

1.3 Aufgabenstellung

Mit Hilfe der zur Verfuegung gestellten VM wird ein vorkonfiguriertes LDAP Service zur Verfügung gestellt. Dieser Verzeichnisdienst soll verwendet werden, um den PublicKey von einem Service zu veröffentlichen. Der PublicKey wird beim Start des Services erzeugt und im LDAP Verzeichnis abgespeichert. Wenn der Client das Service nutzen will, so muss zunaechst der PublicKey des Services aus dem Verzeichnis gelesen werden. Dieser PublicKey wird dazu verwendet, um den symmetrischen Schluessel des Clients zu verschlüsseln und im Anschluss an das Service zu senden. Das Service empfängt den verschlüsselten symmetrischen Schluessel und entschlüsselt diesen mit dem PrivateKey. Nun kann eine Nachricht verschlüsselt mit dem symmetrischen Schlüssel vom Service zum Client gesendet werden.
Der Client empfängt die verschlüsselte Nachricht und entschlüsselt diese mit dem symmetrischen Schlüssel. Die Nachricht wird zuletzt zur Kontrolle ausgegeben.

Die folgende Grafik soll den Vorgang verdeutlichen:

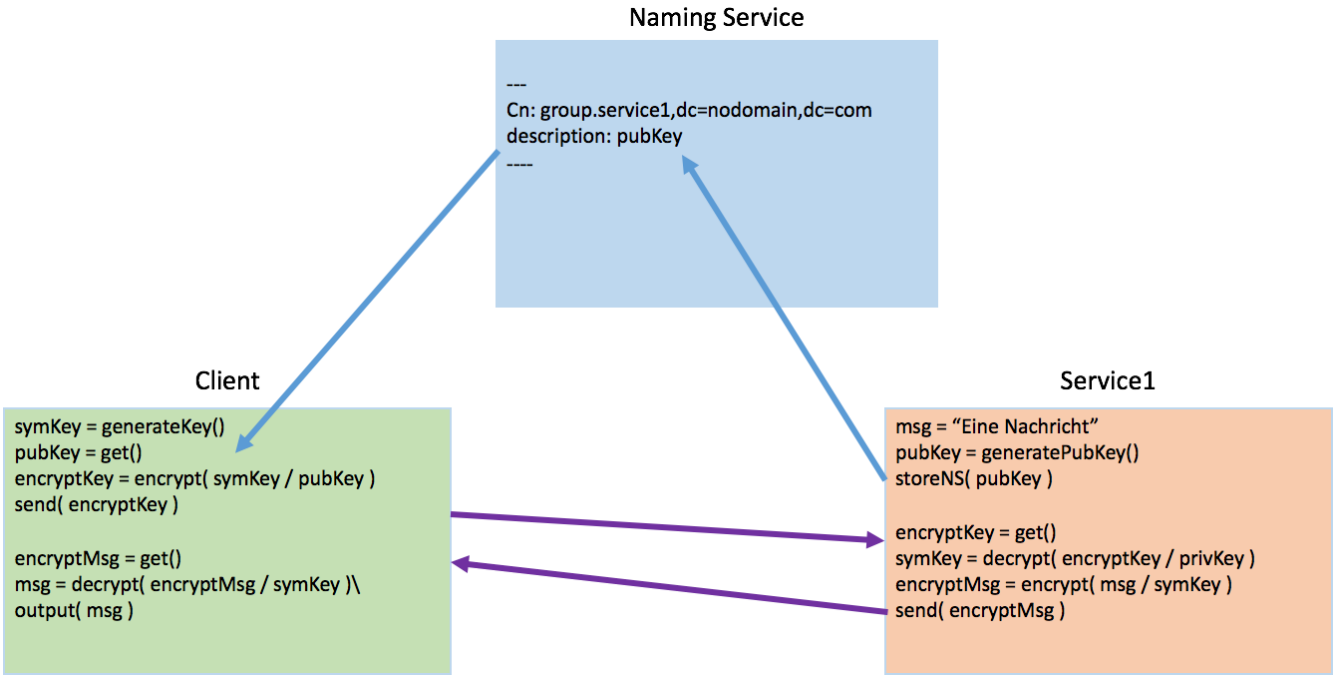

Abbildung 1: Angabe

1.4 Bewertung

16 Punkte

- asymmetrische Verschluesselung (4 Punkte)
- symmetrische Verschluesselung (4 Punkte)
- Kommunikation in Java (3 Punkte)
- Verwendung eines Naming Service, JNDI (3 Punkte)
- Protokoll (2 Punkte)

2 Ergebnisse

Mit folgenden Klassen kann diese Übung erfolgreich abgeschlossen werden:

- KeyPairGenerator
- SecureRandom
- KeyFactory
- X509EncodedKeySpec
- Cipher

Der Ablauf kann mit folgenden Schritten beschrieben werden:

- Generieren eines asymmetrischen Schlüssels, Public Key
- Abspeichern des Public Keys am LDAP-Server
- Client generiert einen symmetrischen Schlüssel
- Public Key wird vom Client aus dem LDAP-Servereintrag geholt und gespeichert
- Der symmetrische Schlüssel wird mit dem Public Key verschlüsselt
- Dieser Schlüssel wird per Socket zum Service gesendet
- Service entschlüsselt empfangenen Key mithilfe seines Private Keys
- Erstellen einer Nachricht vom Service, verschlüsseln mit symmetrischem Schlüssel
- Client empfängt Nachricht ung entschlüsselt diese wiederum mit dem vorher generierten symmetrischen Schlüssel
- Ausgeben der Nachricht durch den Client

Folgende Klassen wurden erstellt:

- Client
- CommunicationClient
- CommunicationServer
- JavaSecurity
- LDAPConnector
- Service1

Die Grundfunktionalität wurde bereits oben erläutert (Client/Service1). CommunicationClient verbindet sich zu einem Server-Socket, kann Byte-Arrays versenden. CommunicationServer öffnet einen Server-Socket auf dem angegebenen Port, auch hier können Byte-Arrays gesendet werden. Die Sendevorgänge setzen eine vorherige Übermittlung der Arraygröße voraus. Die Byte-/Hex-Transition wurde bestehenden Methoden entnommen. Es ist auch zu beachten, dass alle generierten Schlüssel festgeschriebene Header und Footer haben. Der Ablauf auf Codeebene ist dem Quelltext zu entnehmen.

2.1 Symmetrische Verschlüsselung, AES

Die Klasse KeyGenerator stellt hierfür die geeignete Funktionalität bereit.

```
1 KeyGenerator keygen = KeyGenerator.getInstance("AES");
  return keygen.generateKey();
```

Listing 1: Symmetrische Verschlüsselung

2.2 Asymmetrische Verschlüsselung, RSA

Auch hier gibt es bereits bestehenden Code, und zwar in der Klasse KeyPairGenerator. Man kann die Schlüssellänge festlegen sowie auch die Zufallswerte, die in die Erstellung einfließen.

```
  KeyPairGenerator generator = KeyPairGenerator.getInstance("RSA");
  SecureRandom values = SecureRandom.getInstance("SHA1PRNG", "SUN");
3 generator.initialize(4096, values);
  return generator.generateKeyPair();
```

Listing 2: Symmetrische Verschlüsselung

Listings

Abbildungsverzeichnis