Lab07 – JSSE

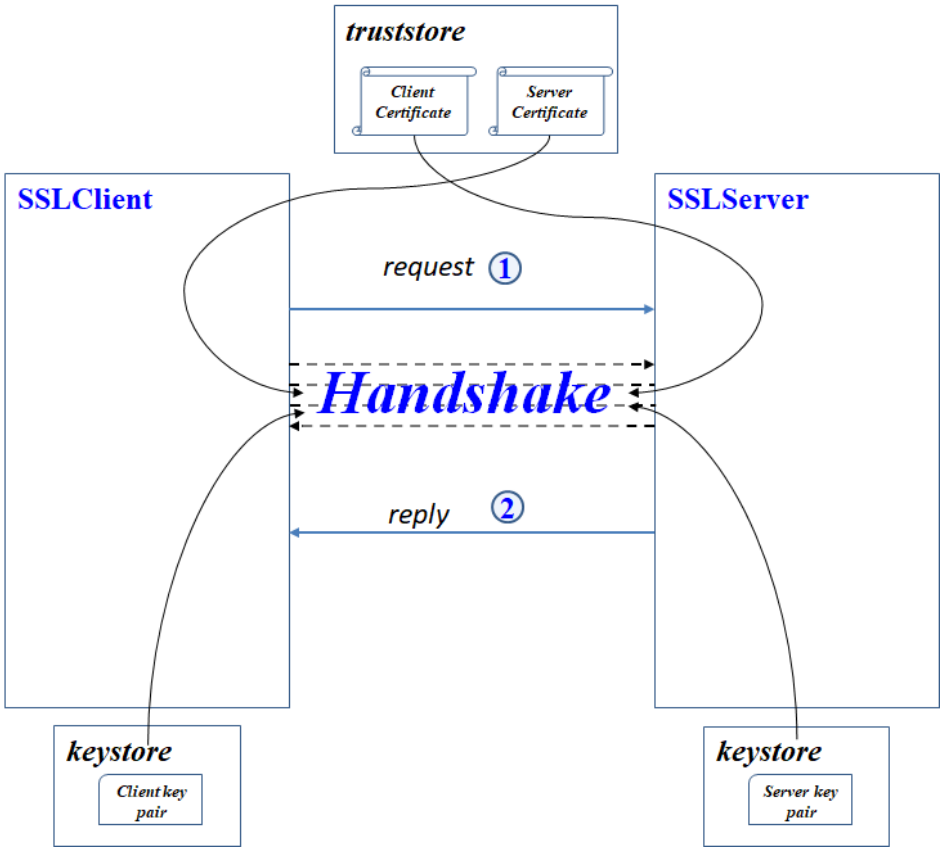
1. Objectivos

O objetivo da 7ª aula TP é familiarizar-vos com a tecnologia JSSE.

Esta tecnologia emprega o protocolo SSL (ou TLS) para estabelecer ligações TCP seguras, ou seja, ligações onde é garantida, dentro de certos limites, a confidencialidade, autenticidade e integridade das informações trocadas.

Para o estabelecimento de uma ligação TCP protegida por SSL, é desenvolvido o seguinte procedimento:

- estabelecimento de uma ligação TCP normal;
- combinação entre as duas partes do conjunto de métodos criptográficos a empregar daí em diante. Este processo chama-se *Handshake* e decorre de forma transparente ao programador;
- continuação da comunicação, agora protegida pelos métodos antes acordados.



2. Classes mais Relevantes

As classes Java a empregar para o estabelecimento de uma comunicação TCP segura são *SSLServerSocket* e *SSLSocket* da package *javax.net.ssl*. A forma de as empregar é em tudo semelhante ao estabelecimento de uma comunicação TCP não segura. As diferenças estão:

- na geração dos *sockets*;
- e nas preparações a montante, do estabelecimento da ligação, que estão relacionadas com o processo de *handshake*.

No que respeita à geração da *sockets*, esta requer o emprego de uma *factory* (*SSLServerSocketFactory* ou *SSLSocketFactory* da package *javax.net.ssl*) para criar instancias de *SSLServerSocket* e *SSLSocket* respectivamente.

```
SSLServerSocket s = null;
SSLServerSocketFactory ssf = null;

ssf = (SSLServerSocketFactory) SSLServerSocketFactory.getDefault();

try
{
    s = (SSLServerSocket) ssf.createServerSocket(port);
}
catch( IOException e)
{
    System.out.println("Server - Failed to create SSLServerSocket");
    e.getMessage();
    return;
}
```

No lado cliente a situação é semelhante, sendo necessário usar as classes correspondentes a *SSLSocket* em vez de *SSLServerSocket*.

De forma semelhante à classe *ServerSocket*, o método *accept*() da classe *SSLServerSocket* retorna um objecto do tipo *SSLSocket*, que pode então ser usado para comunicação segura.

3. *Handshake*

O processo de *handshake* é o mecanismo através do qual cliente e servidor combinam as ferramentas criptográficas a empregar para garantir a segurança do canal de comunicação entre eles. Esta segurança implica o emprego de ferramentas de criptografia simétrica e assimétrica.

Na criptografia simétrica, a mesma chave, secreta, é conhecida pelas duas partes comunicantes e é empregue para cifrar e decifrar as mensagens trocadas.
Na criptografia assimétrica as duas partes possuem uma chave dupla, composta por uma parte secreta (chave privada) e uma parte pública (chave pública) que deve ser efectivamente do conhecimento público. Um conteúdo que seja cifrado com uma das partes da chave só pode ser decifrado com a outra parte e vice-versa.
Um conteúdo que seja cifrado com a chave privada só pode ser decifrado com a respectiva chave pública. Isto é assim equivalente a uma assinatura, pois só pode ser feita pelo próprio (que detém a chave privada secreta) e pode ser verificada por todos (pois todos conhecem a chave pública respectiva).

Um conteúdo cifrado com a chave pública só pode ser decifrado pelo detentor da chave privada (secreta). Isto funciona assim como uma cifragem de um conteúdo (que qualquer parte pode fazer) que só pode ser decifrado pelo detentor da chave privada.

O desenvolvimento do processo *handshake* requer que ele próprio seja seguro. É assim necessário que as duas entidades comunicantes se autentiquem uma perante a outra (tipicamente apenas o lado servidor se autêntica perante o cliente, ex. HTTPs), e que a comunicação entre os dois seja confidencial, num momento em que ainda não existe qualquer segredo partilhado entre os dois lados.

Para alcançar tal objectivo na fase inicial do processo de *handshake* é empregue criptografia assimétrica para garantir a autenticidade das mensagens (assinaturas com chave privada do emissor) e a confidencialidade das mesmas (cifragem com a chave pública do receptor). Usando este mecanismo as duas partes acordam as ferramentas criptográficas simétricas a empregar posteriormente e, daí para a frente, a comunicação passa a desenvolver-se sob a protecção de tais ferramentas simétricas.

Este processo implica a existência de mecanismos que permitam o armazenamento seguro de chaves privadas e a publicação confiável de chaves públicas. Implica assim que:

- cada entidade deve ter uma ***keystore*** encriptada onde guarda o seu par de chaves criptográficas assimétricas;
- cada interlocutor deverá possuir um certificado onde é expressa a sua chave pública. Este certificado é assinado (com a chave privada) por uma autoridade superior, ou (como no caso deste lab) pelo próprio interlocutor (o cliente ou o servidor). Estes certificados devem ser guardados em estruturas chamadas de ***truststores*** e estas devem ser disponibilizadas publicamente.

Assim, acedendo à ***truststore*** (pública e conhecida de todos os intervenientes) e obtendo o certificado do nosso interlocutor podemos obter a sua chave pública e validar a autenticidade das suas mensagens (que ele assina com a sua chave privada) vice-versa.

4. Configuração

A montante do processo de estabelecimento da comunicação segura TCP é necessário indicar à JVM (na linha de comandos aquando do *startup*, ou programaticamente) as localizações da ***truststore*** e ***keystore*** **e** da chave de encriptação da ***keystore*** (tipicamente o cliente precisa de conhecer a ***truststore*** e o servidor precisa de conhecer a ***keystore*** e sua chave de encriptação).

```
//set the name of the trust store containing the server's public key and certificate
System.setProperty("javax.net.ssl.trustStore", "truststore");

//set the type of trust store
System.setProperty("javax.net.ssl.trustStoreType","JKS");

//set the password with which the keystore is encripted
System.setProperty("javax.net.ssl.trustStorePassword", "123456");
```

No caso do lab de SDIS sobre JSSE vocês precisarão então de gerar pares de chaves, construir uma ***truststore*** e uma ***keystore***.

Para perceberem como o podem fazer devem procurar no tutorial JSSE da Oracle (secção *Creating a Keystore to Use with JSSE*) indicado no fim da página com o guião do lab.