Trabalho 1 – Serviço de Backup Distribuído

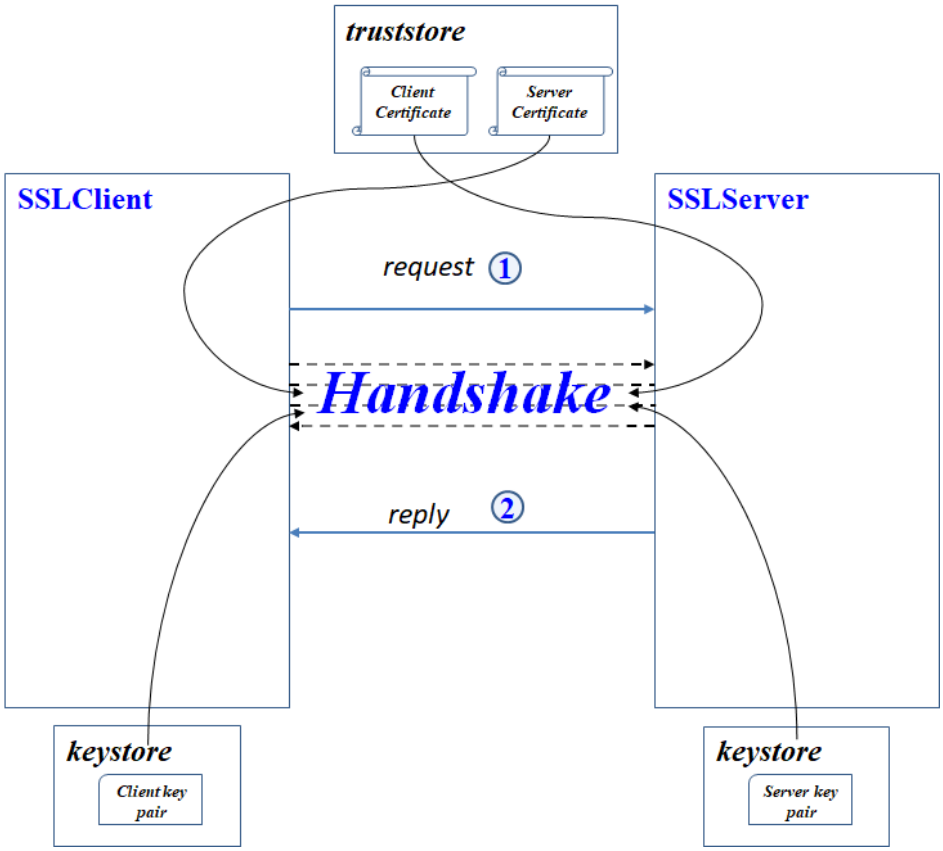
1. Objectivos para a Aula 9

O objetivo da 9ª 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



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 *ServerSocket* 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.

No que respeita às preparações para o *handshake*: para que o processo de *handshake* seja ele próprio seguro, é necessário que as duas entidades comunicantes se autentiquem uma perante a outra (tipicamente apenas o lado servidor se autêntica perante o cliente, ex. HTTPs).

Para isso é empregue um sistema de chaves assimétricas e de certificados assinados que garantem a identidade de cada interlocutor.

- Chaves assimétricas – cada entidade deve possuir um par de chaves criptográficas. Uma pública e outra privada. Um conteúdo cifrado com a chave pública só pode ser decifrado com a privada e vice-versa. Assim se uma entidade cifra algo com a sua chave privada, esse conteúdo pode ser decifrado apenas com a chave pública correspondente demonstrando assim que quem cifrou o conteúdo detêm a chave privada correspondente e assim é quem afirma ser. Cada entidade deve guardar as suas chaves numa ***keystore*** encriptada;
- Certificados – 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.

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");
```

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*) indicados no fim da página com o guião do lab.