First we need to make certificate, this is done by using keytool that is part of J2SE SDK (program will ask for certificate owner information and password, enter 123456 as password, or you can enter your password, but notice that you have to change it in other commands listen in this tutorial):

**keytool -genkey -keystore mySrvKeystore -keyalg RSA**

After this command you will have certificate file in working directory of issuing keytool command

Server source code (EchoServer.java)

import javax.net.ssl.SSLServerSocket;

import javax.net.ssl.SSLServerSocketFactory;

import javax.net.ssl.SSLSocket;

import java.io.BufferedReader;

import java.io.InputStream;

import java.io.InputStreamReader;

public

class EchoServer {

public

static

void

main(String[] arstring) {

try {

SSLServerSocketFactory sslserversocketfactory =

(SSLServerSocketFactory) SSLServerSocketFactory.getDefault();

SSLServerSocket sslserversocket =

(SSLServerSocket) sslserversocketfactory.createServerSocket(9999);

SSLSocket sslsocket = (SSLSocket) sslserversocket.accept();

InputStream inputstream = sslsocket.getInputStream();

InputStreamReader inputstreamreader = new InputStreamReader(inputstream);

BufferedReader bufferedreader = new BufferedReader(inputstreamreader);

String string = null;

while ((string = bufferedreader.readLine()) != null) {

System.out.println(string);

System.out.flush();

}

} catch (Exception exception) {

exception.printStackTrace();

}

}

}

Compile it by using simple command:

Client source code (EchoClient.java)

import javax.net.ssl.SSLSocket;

import javax.net.ssl.SSLSocketFactory;

import java.io.\*;

public

class EchoClient {

public

static

void

main(String[] arstring) {

try {

SSLSocketFactory sslsocketfactory = (SSLSocketFactory) SSLSocketFactory.getDefault();

SSLSocket sslsocket = (SSLSocket) sslsocketfactory.createSocket("localhost", 9999);

InputStream inputstream = System.in;

InputStreamReader inputstreamreader = new InputStreamReader(inputstream);

BufferedReader bufferedreader = new BufferedReader(inputstreamreader);

OutputStream outputstream = sslsocket.getOutputStream();

OutputStreamWriter outputstreamwriter = new OutputStreamWriter(outputstream);

BufferedWriter bufferedwriter = new BufferedWriter(outputstreamwriter);

String string = null;

while ((string = bufferedreader.readLine()) != null) {

bufferedwriter.write(string + '\n');

bufferedwriter.flush();

}

} catch (Exception exception) {

exception.printStackTrace();

}

}

}

Compile it by using simple command:

javac EchoClient.java

# Running server and client using SSL

First copy certificate file that you created before into working directory and run server with these parameters (notice that you have to change keyStore name and/or trustStrorePassword if you specified different options creating certificate:

java -Djavax.net.ssl.keyStore=mySrvKeystore -Djavax.net.ssl.keyStorePassword=123456 EchoServer

And now again copy certificate file that you created before into working directory and run client with these parameters (notice that you have to change keyStore name and/or trustStrorePassword if you specified different options creating certificate:

java -Djavax.net.ssl.trustStore=mySrvKeystore -Djavax.net.ssl.trustStorePassword=123456 EchoClient

If you want **SSL** debug information just add these parameters when running server and/or client:

-Djava.protocol.handler.pkgs=com.sun.net.ssl.internal.www.protocol -Djavax.net.debug=ssl

# Playing with server and client

Now just type any string on client console and press return. The same string has to appear on server console.

SSL (Secure Sockets Layer) is the standard security technology for establishing an encrypted link between a web server and a browser. This link ensures that all data passed between the web server and browsers remain private and integral. SSL is an industry standard and is used by millions of websites in the protection of their online transactions with their customers.

To be able to create an SSL connection a web server requires an SSL Certificate. When you choose to activate SSL on your web server you will be prompted to complete a number of questions about the identity of your website and your company. Your web server then creates two cryptographic keys - a Private Key and a Public Key.

The Public Key does not need to be secret and is placed into a Certificate Signing Request (CSR) - a data file also containing your details. You should then submit the CSR. During the SSL Certificate application process, the Certification Authority will validate your details and issue an SSL Certificate containing your details and allowing you to use SSL. Your web server will match your issued SSL Certificate to your Private Key. Your web server will then be able to establish an encrypted link between the website and your customer's web browser.

The complexities of the SSL protocol remain invisible to your customers. Instead their browsers provide them with a key indicator to let them know they are currently protected by an SSL encrypted session - the lock icon in the lower right-hand corner, clicking on the lock icon displays your SSL Certificate and the details about it. All SSL Certificates are issued to either companies or legally accountable individuals.

Typically an SSL Certificate will contain your domain name, your company name, your address, your city, your state and your country. It will also contain the expiration date of the Certificate and details of the Certification Authority responsible for the issuance of the Certificate. When a browser connects to a secure site it will retrieve the site's SSL Certificate and check that it has not expired, it has been issued by a Certification Authority the browser trusts, and that it is being used by the website for which it has been issued. If it fails on any one of these checks the browser will display a warning to the end user letting them know that the site is not secured by SSL.

What is Secure Sockets Layer (SSL)?

Secure Sockets Layer (SSL) is a standard security technology for establishing an encrypted link between a server and a client—typically a web server (website) and a browser, or a mail server and a mail client (e.g., Outlook).

SSL allows sensitive information such as credit card numbers, social security numbers, and login credentials to be transmitted securely. Normally, data sent between browsers and web servers is sent in plain text—leaving you vulnerable to eavesdropping. If an attacker is able to intercept all data being sent between a browser and a web server, they can see and use that information.

More specifically, SSL is a security protocol. Protocols describe how algorithms should be used. In this case, the SSL protocol determines variables of the encryption for both the link and the data being transmitted.

All browsers have the capability to interact with secured web servers using the SSL protocol. However, the browser and the server need what is called an SSL Certificate to be able to establish a secure connection.

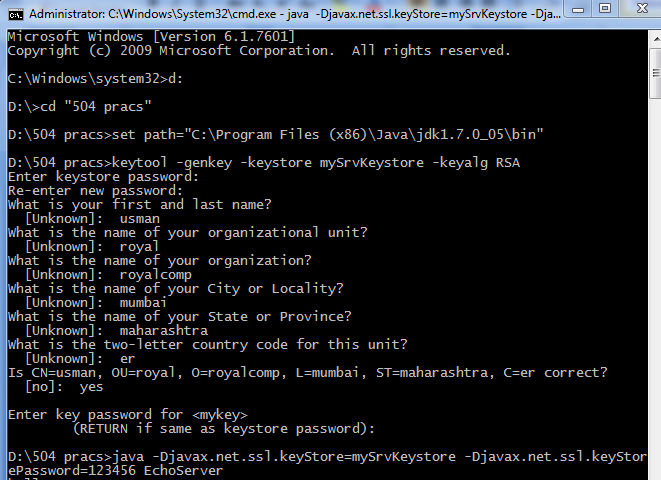
SSL secures millions of peoples’ data on the Internet every day, especially during online transactions or when transmitting confidential information. Internet users have come to associate their online security with the lock icon that comes with an SSL-secured website or green address bar that comes with an Extended Validation SSL-secured website. SSL-secured websites also begin with https rather than http.

STEP1: do this in server…

**>set path**

**>keytool -genkey -keystore mySrvKeystore -keyalg RSA**

**>** **java -Djavax.net.ssl.keyStore=mySrvKeystore -Djavax.net.ssl.keyStorePassword=123456 EchoServer**

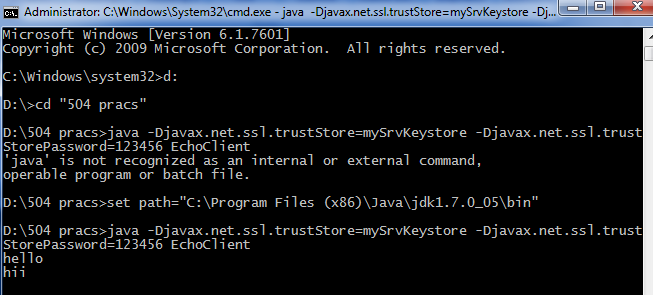


STEP2: do this(say hello) in client…

**>set path**

**>** **java -Djavax.net.ssl.trustStore=mySrvKeystore -Djavax.net.ssl.trustStorePassword=123456 EchoClient**

**(---send some messages---)**



STEP3: you’ll see this(output) in server…