

# Secure Authenticated Java Communication with JSSE

RJUG May 11, 2004

Ralph Durkee,  
Security Certs: CISSP, GSEC, GCIH

[www.rd1.net](http://www.rd1.net) [rd@rd1.net](mailto:rd@rd1.net)

# Outline

---

- # SSL Protocol and Certificates
- # JSSE Architecture and Implementation
- # Java and SSL on-line Resources
- # Certificate management
- # SSL Server keys
- # Certificate signing
- # SSL Client keys and Installation
- # Authenticated Client example code with JSSE

# SSL Secure Sockets Layer Overview

## Provides

- # Server Authentication
- # Client Authentication
- # Encryption of Communication
- # Integrity of Communication
- # Used along with any application level protocol such as HTTP
- # HTTP over SSL = HTTPS
- # POP3 over SSL = POP3S a.k.a. SPOP3

# Terms

- # TLS – new IETF name for 3.x and newer versions of SSL (TLS 1.0 = SSL 3.1)
- # JSSE – Java Secure Sockets Extensions reference implementation provided by Sun.
- # Certificate -- a digitally signed statement vouching for the identity of a person or company.
- # Works like a public key for asymmetric encryption.
- # For SSLv3 Protocol details  
<http://wp.netscape.com/eng/ssl3/3-SPEC.HTM>

# SSL Handshake (simplified) With Client Certificate

**Client Sends →**

**← Server Replies**

1. Client hello

2. Server hello

3. Server Cert. & CAs

4. Cert. Request

5. Key Exchange

6. Server hello done

7. Client Cert.

# SSL Handshake (continued)

**Client Sends →**

8. Client key exchange
9. Certificate verify
10. Change cipher spec
11. Finished

14. Encrypted data

**← Server Replies**

12. Change cipher spec

13. Finished

14. Encrypted data

# Example Certificate Details

- # **Owner:** CN=ssl.rjug.org, OU=Web Development, O=Rochester Java Users Group, L=Rochester, ST=New York, C=US
- # **Issuer:** EmailAddress=webmaster@rd1.net, CN=www.rd1.net, OU=Internet Development, O=Ralph Durkee Consulting, L=Lima, ST=New York, C=US
- # **Valid from:** Fri Sep 21 12:51:00 EDT 2003 until: Sat Sep 21 12:51:00 EDT 2004

# PEM or RFC or Base64 Format

-----BEGIN CERTIFICATE-----

```
MIICyTCCAjKgAwIBAgIDD8LoMA0GCSqGSIb3DQEBAgUAMIHEMQswCQYDVQQGEwJa
QTEVMBMGA1UECBMMV2VzdGVybiBDYXBlMRIwEAYDVQQHEw1DYXBlIFRvd24xHTAB
BgNVBAoTFFRoYXd0ZSBDb25zdWx0aw5nIGNjMSgwJgYDVQQLEx9DZXJ0awZpY2F0
aw9uIFNlcnZpY2VzIERpdmlzaW9uMRkwFwYDVQQDEXBuGF3dGUgU2VydmVyIENB
MSYwJAYJKoZIhvcNAQkBFhdzZXJ2ZXItY2VydHNAAdGhhd3RlLmNvbTAeFw0wMTA3
MTMxNjExMjNaFw0wMzA4MDIxOTI4MThaMGcxGzAJBgNVBAYTA1VTMREwDwYDVQQI
Ewh0ZXcgWw9yazENMA5GA1UEBxMETGltYTEgMB4GA1UEChMXUmFscGggRHVya2Vl
IENvbnN1bHRpbmcxFDASBgNVBAMTC3NzbC5yZDEubmV0MIGfMA0GCSqGSIb3DQEB
AQUAA4GNADCBiQKBgQC2lNe5tV6BVBthPr1Cxc1yfUKdT4lFY4SpP00S+q50DzS4
0KhhcBkwd/rT01WS34d3g+wQwQZUM1bL98D/gmsr+XZqLL1h5l02hRr02gwc9kDY
FwzMudn1U0ezkrbbYVrBnMwZZ1VDXLWbB7npZsz59JsoIV4gnUSxxTMMMykmewID
AQABoyUwIzATBgNVHSUEDDAKBggrBgEFBQcDATAMBgNVHRMBAf8EAjAAMA0GCSqG
SIb3DQEBAgUAA4GBAGLYzpGgn88I5SQ6pgSUFBV5SPwZhQUuW51PkHLCiusJpxM6
iZwSjufpAtL8HG8ppq99noD5x6F/Q/0v4Xcwr6b7pxDl8ay4+mhGZDI1Rj1q4Vit4
GndEqTB8iKFHmjC9YDLVmkxYDof8/tqMZz16w/ynFvemEp9o8doq9aqPu6A2
```

-----END CERTIFICATE-----



# JSSE Architecture

---

## JSSE Packages

- # **javax.net.ssl** -- Provides the classes for the secure socket optional package.
- # **javax.net** -- Provides optional classes for networking applications.
- # **javax.security.cert** -- Provides optional classes for public key certificates.

# JSSE Architecture

## 2 API's provided for SSL Applications

### 1. Socket and Certificate Level

- Direct usage of SSL Sockets
- Programmed Certificate management

### 1. URL Level

- URLConnection class
- Just say “https:// . . .”
- Most existing code need not change!

# Package javax.net.ssl

---

# provides SSL Socket level API

# Classes

- SSLServerSocket
- SSLServerSocketFactory
- SSLSocket
- SSLSocketFactory

# Java & SSL on-line Resources

---

## # JSSE Reference Guide

<http://java.sun.com/j2se/1.4.2/docs/guide/security/jsse/JSSERefGuide.html>

## # Intro. to JSSE at Java One 2003

<http://java.sun.com/security/javaone/2003/2226-JSSE.pdf>

## # OpenSSL (Use for generating server certificates)

<http://www.openssl.org/>

## # JSSE download needed for the old JDK 1.3

<http://java.sun.com/products/jsse/>

# Java Keytool

## # Contains multiple certificates and keys.

- Certificate = public & key = private

## # Provides Java Certificate management

- **-help** provide usage summary
- **-list** will list certificates in a keystore
- **-export** exports a single certificate to a file.
- **-printcert** shows certificate details

## # on-line

<http://java.sun.com/j2se/1.4.2/docs/tooldocs/tools.html>

# Keytool printcert example

```
$ keytool -printcert -file client1.crt
```

```
Owner: CN=client1.rjug.org, OU=Educational Services, O="Durkee Consulting, Inc.", L=Lima, ST=New York, C=US
```

```
Issuer: EMAILADDRESS=hostmaster@rd1.net, CN=Durkee Certificate Authority, OU=Internet Services, O="Durkee Consulting, Inc.", L=Lima, ST=NY, C=US
```

```
Serial number: 1
```

```
Valid from: Tue May 11 10:24:52 EDT 2004 until: Wed May 11 10:24:52 EDT 2005
```

```
Certificate fingerprints:
```

```
MD5: 5F:12:DF:1E:B1:83:58:53:F8:68:CF:E7:AC:09:42:8C
```

```
SHA1:
```

```
FB:12:66:10:1E:E1:DC:45:5C:F2:32:7F:A6:A1:DC:3B:5F:44:F8:17
```

# Server Key Generation

## Generate Server Key Using openssl

<http://www.openssl.org>

```
# Example commands using Unix sh, ksh or bash shell
# Setup env and path
OPENSSL="/usr/local/ssl/"
PATH="$OPENSSL/bin/:$PATH"
# Generate an RSA private key for your server
# Remember the pass phrase and backup the key securely.
# passphrase should be long with special characters.
openssl genrsa -des3 -out ssl.rjug.org.key 2048
```

# Server Key Format

- # The output is an encrypted in RFC Base64 format

```
-----BEGIN RSA PRIVATE KEY-----
```

```
Proc-Type: 4, ENCRYPTED
```

```
DEK-Info: DES-EDE3-CBC, 8F460A4AE271B821
```

```
QGc8gU00z4ZHGKV1WzgVoItvc3NYTVV7qMr8lCxmAJOb+NWEU . . .
```

```
7WFxdIqGheZV1r0yFU08gAsd1HoJ/HZdaIYebMZVw/0txo/
```

```
. . .
```

```
6Mi87XhfYot9I2Kwzymw1SpEiVMwPxu0c+ISY1PmzB0H4eueZ4g==
```

```
-----END RSA PRIVATE KEY-----
```



# Certificate Signing Request (CSR) Generation

```
openssl req -new -key ssl.rjug.org.key -out ssl.rjug.org.csr
Enter pass phrase for ssl.rjug.org.key: (passwd doesn't echo)
You are about to be asked to enter information that will be
  incorporated into your certificate request.
What you are about to enter is what is called a Distinguished
  Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [US]:
State or Province Name (full name) [New York]:
Locality Name (eg, city) [Lima]: Rochester
```

# Certificate Signing Request Generation Continued

Organization Name (eg, company) [Durkee Consulting, Inc.]: **Rochester Java Users Group**

Organizational Unit Name (eg, section) []:

Common Name (eg, YOUR name) []: **ssl.rjug.org**

Email Address []: **rjug@rjug.org**

Please enter the following 'extra' attributes  
to be sent with your certificate request

A challenge password []:

An optional company name []:

**#** Note: The *Common Name* must be the FQDN of the URL  
for accessing the server.

# Format of the Certificate Signing Request

## Format of Certificate Request

```
$> more ssl.rjug.org.csr
```

```
-----BEGIN NEW CERTIFICATE REQUEST-----
```

```
MIIB4zCCAUwCAQAwwgIx CzA JBgNVBAYTA1VTMREwDwYDVQQIEwhO...
```

```
MA8GA1UEBxMIRmFpcnBvcnQxJzA1BgNVBAoTH1RBUkdVUyBJbmZvc...
```

```
...
```

```
...VnpP3PzVcSFKeFYR4qpzc0w6Bcj19AszHkc5q/x8
```

```
/ECIDP07Gw==
```

```
-----END NEW CERTIFICATE REQUEST-----
```

Or have your Certificate Signing Request signed by a Certificate Authority such as Verisign or Thawte, if you need built-in browser acceptance.

Work with your hosting provider such as **rd1.net** (Durkee Consulting) to get a sign your certificate.

Or set up your own Certificate Authority.

# Format of the Signed Server Certificate

# The Signed Certificate will be returned by the Certificate Authority.

# Example Format:

```
-----BEGIN CERTIFICATE-----  
MIID4zCCA0ygAwIBAgIBBjANBgkqhkiG9w0BAQQFADCBqDELMAkGA1U . . .  
ETAPBgNVBAGTCE5ldyBZb3JrMQ0wCwYDVQQHEwRMaW1hMSAwHgYDVQQK  
. . .  
Yn2n3Ba7MYhhZzSYkjenMjvln4ql2g/Wicyh77lonPs4VSPrn  
-----END CERTIFICATE-----
```

# Apache Configuration for Server Certificate

## Example httpd.conf Configuration:

```
<VirtualHost ssl.rjug.org:443>
#   SSL Engine Switch:
SSLEngine on
#   Server Certificate:
SSLCertificateFile /etc/cert/ssl.rd1.net.crt
#   Server Private Key:
SSLCertificateKeyFile /etc/cert/ssl.rd1.net.key
#   Server Certificate Chain:
SSLCertificateChainFile /etc/cert/dcica.crt
#   SSL Protocol and Cipher Suites:
SSLProtocol all -SSLv2
SSLCipherSuite 'ALL:!SSLv2:!LOW:!ADH'
```

# Apache Configuration To Require Client Certificates

## Example httpd.conf Configuration:

```
# Certificate Authority (CA):  
SSLCACertificateFile /etc/cert/dcica.crt  
# Certificate Revocation Lists (CRL): (optional)  
#SSLCARevocationPath /etc/apache/ssl.crl  
# Client Authentication (Type):  
# Types are none, optional, require and optional_no_ca.  
# Depth specifies how deeply to verify the certificate  
# issuer chain before deciding the certificate is not valid.  
SSLVerifyClient require  
SSLVerifyDepth 10
```

# Java Client Key Generation

- # Use Java keytool recommended
- # Any platform (MS win32 example)

```
path=C:\jdk1.3\bin;%PATH%
```

```
C:\>keytool -genkey -keystore clientstore -alias  
client1 -storepass use-your-own-password
```

```
What is your first and last name?
```

```
[Unknown]: client1.rjug.org
```

```
What is the name of your organizational unit?
```

```
[Unknown]: Educational Services
```

# Java Client Key Generation

## Continued

What is the name of your organization?

[Unknown]: **Durkee Consulting, Inc.**

What is the name of your City or Locality?

[Unknown]: **Lima**

What is the name of your State or Province?

[Unknown]: **New York**

What is the two-letter country code for this unit?

[Unknown]: **US**

Is <CN= **client1.rjug.org**, OU= **Educational Services**, O= **Customer Corporation**, L= **Anywhere**, ST=New York, C=US> correct? [no]: **yes**

Enter key password for <client1>

(RETURN if same as keystore password):



# Client KeyStore

## # Listing the KeyStore

```
$ keytool -list -keystore clientstore
```

```
Enter keystore password: use-your-own-password
```

```
Keystore type: jks
```

```
Keystore provider: SUN
```

```
Your keystore contains 1 entry:
```

```
client1, May 11, 2004, keyEntry,
```

```
Certificate fingerprint (MD5):
```

```
DC:80:90:69:20:99:AC:50:81:AF:BC:5A:A9:3C:4C:3D
```

# Client Certificate Signing Request

## # Generate CSR with keytool

```
$ keytool -certreq -keystore clientstore -alias client1 -file client1.csr
```

```
Enter keystore password: use-your-own-password
```

```
$ more client1.csr
```

```
-----BEGIN NEW CERTIFICATE REQUEST-----
```

```
MIICKTCCAk8CAQAwwYsxCzAJBgNVBAYTA1VTRREwDwYDVQQQIEwhOZXcgW9yaz  
ENMAsGA1UEBxME
```

```
. . .
```

```
RQKs6s1IzP8QV5JBAhQD80J9qYZ/tzvVEPiY7UFVANIZDg==
```

```
-----END NEW CERTIFICATE REQUEST-----
```

## # Send CSR to Certificate Authority (CA) to be Signed.

## # Receive client1.crt from CA

# CA Signing the Client Certificate

## Use OpenSSL Certificate Authority

```
$ openssl ca -policy policy_anything -out client1.crt  
-infile client1.csr
```

```
Using configuration from /etc/ssl/openssl.cnf
```

```
Enter pass phrase for /etc/dciCA/private/dcica.key:
```

```
Check that the request matches the signature
```

```
Signature ok
```

```
Certificate Details:
```

```
Serial Number: 16 (0x10)
```

```
Validity
```

```
Not Before: May 11 14:24:52 2004 GMT
```

```
Not After : May 11 14:24:52 2005 GMT
```

# CA Signing the Client Certificate (2)

## Subject:

countryName	=	<b>US</b>
stateOrProvinceName	=	<b>New York</b>
localityName	=	<b>Lima</b>
organizationName	=	<b>Durkee Consulting, Inc.</b>
organizationalUnitName	=	<b>Educational Services</b>
commonName	=	<b>client1.rjug.org</b>

## X509v3 extensions:

### X509v3 Basic Constraints:

CA:FALSE

### Netscape Comment:

OpenSSL Generated Certificate

### X509v3 Subject Key Identifier:

F2:EC:B2:47:5E:C3:ED:C5:2A:20:7B:03: . . .

# CA Signing the Client Certificate (3)

```
DirName:/C=US/ST=NY/L=Lima/O=Durkee Consulting,  
Inc./OU=Internet
```

```
Services/CN=Durkee Certificate
```

```
Authority/emailAddress=hostmaster@rd1.net
```

```
serial:00
```

```
Certificate is to be certified until May 11 14:24:52 2005  
GMT (365 days)
```

```
Sign the certificate? [y/n]:y
```

```
1 out of 1 certificate requests certified, commit? [y/n]y
```

```
Write out database with 1 new entries
```

```
Data Base Updated
```

May 11, 2004

<http://rd1.net>

# Import CA Certificate

```
C:/> keytool -import -file dcica.crt -alias dcica -keystore
clientstore -trustcacerts
Enter keystore password: use-your-own-password
Owner: EMAILADDRESS=hostmaster@rd1.net, CN=Durkee Certificate
Authority, OU=Internet Services, O="Durkee Consulting,
Inc.", L=Lima, ST=NY, C=US
Issuer: EMAILADDRESS=hostmaster@rd1.net, CN=Durkee Certificate
Authority, OU=Internet Services, O="Durkee Consulting,
Inc.", L=Lima, ST=NY, C=US
Serial number: 0
Valid from: Fri Jan 23 16:33:53 EST 2004 until: Sat Jan 22
16:33:53 EST 2005
Certificate fingerprints:
    MD5: CB:43:1A:91:37:0C:24:A2:33:D1:46:1A:66:8B:A6:87
    SHA1:
    0F:1E:23:C4:95:7E:AD:34:55:2B:66:06:66:91:3C:B2:09:64:1B:A7
Trust this certificate? [no]: yes
Certificate was added to keystore
May 11, 2004 http://rd1.net
```

# Import Signed Client Certificate

## Import the signed client certificate

```
$ keytool -import -file client1.crt -alias client1  
-keystore clientstore
```

```
Enter keystore password: use-your-own-password
```

```
Certificate reply was installed in keystore
```

```
$ keytool -list -keystore clientstore
```

```
Enter keystore password: use-your-own-password
```

```
Keystore type: jks
```

```
Keystore provider: SUN
```

```
Your keystore contains 2 entries
```

```
client1, May 11, 2004, keyEntry,
```

```
Certificate fingerprint (MD5):
```

```
5F:12:DF:1E:B1:83:58:53:F8:68:CF:E7:AC:09:42:8C
```

```
dcica, May 11, 2004, trustedCertEntry,
```

```
Certificate fingerprint (MD5):
```

```
CB:43:1A:91:37:0C:24:A2:33:D1:46:1A:66:8B:A6:87
```

# Java Client Declarations

```
// Java client to demonstrate secure SSL communication
//   with client certificate based authenticated.
//
// by Ralph Durkee http://rd1.net

import java.net.*;
import java.io.*;
import java.security.Security;

public class client1 {

    public static void main( String[] args ) throws Exception {

        String crt_store = args[0];           // key and trust store
        String urlstr = args[1];             // https:// ...
        String user_name = args[2];          // user name to post
        String passwd = args[3];             // password to post
    }
}
```



# Java Client Setting Properties

```
// Dynamic registration of JSSE provider
// This may also be added to the <java-home>/lib/security/java.security
Security.addProvider( new com.sun.net.ssl.internal.ssl.Provider());

// Setup the handler in the system properties for JSSE
// Sys Props can also be set on the cmd line with the -D option
System.setProperty("java.protocol.handler.pkgs", "com.sun.net.ssl.intern
al.www.protocol");

// Storage and password for client certificates
System.setProperty("javax.net.ssl.keyStore", crt_store);
System.setProperty("javax.net.ssl.keyStorePassword", crt_pswd);

// Storage and password for server certificates to be trusted,
System.setProperty("javax.net.ssl.trustStore", crt_store);
System.setProperty("javax.net.ssl.trustStorePassword", crt_pswd);
```

# Java Client

## Optional Properties

### ☒ SSL Verbose Debug option

```
// Enable debug options: ssl, handshake, and trustmanager  
// System.setProperty("javax.net.debug", "ssl,handshake,trustmanager");
```

### ☒ Proxy Server Settings

```
// The Proxy values can not be changed once they are set except  
// by exiting Java, since they are cached.  
// This was reported as a bug to Sun,  
// but has been declared a necessary feature for performance sake.  
// These values are best set on the cmd line.  
// System.setProperty("https.proxyHost", "192.168.2.6" );  
// System.setProperty("https.proxyPort", "8080" );
```

# Java Client

## Preparing the Connection

```
try {
    URL server_url = new URL( urlstr);
    // Setup the Connection properties. Unlike what it sounds like,
    // the openConnection() does not yet open or establish the connection.
    // It simply creates a URLconnection object.
    URLConnection conn = server_url.openConnection();

    // Input = Get operation, Output = post operation
    conn.setDoOutput(true);
    conn.setUseCaches(false);
    conn.setAllowUserInteraction(false);
    // Set the content type to url encoded
    conn.setRequestProperty("Content-type","application/x-www-form-urlencoded");
    // Write the post variables and values in URL encoded format.
    PrintWriter post_req = new PrintWriter(conn.getOutputStream());
    post_req.print("User=" + user_name + "&");
    post_req.print("Passwd=" + passwd);
    // post_req.flush();
    post_req.close();
}
```

# Java Client

## Make the Connection

```
conn.connect();      // Now we are actually connecting.

// Check the return Status
// We can't use getHeaderFieldKey() for the status,
// probably because it lacks the colon format.
// but status is always the first header (i.e. 0).
String http_stat = conn.getHeaderField(0);
// parse the Response Status header to ensure we got a 200
if ( -1 == http_stat.indexOf(" 200 ") ) {
    String emsg = "HTTP Status = "+ http_stat;
    throw new IOException(emsg);
}
```

# Java Client

## Read the Results

```
int len = conn.getContentLength();
if ( len <= 0) len = 1024;
System.out.println(" length = " + len);
BufferedReader in = new BufferedReader( new InputStreamReader(
conn.getInputStream() ) );
char[] content;
content = new char[ len + 1 ];
int c = in.read( content, 0, len);
System.out.println( content);

} catch (IOException ioex) {
    System.err.println("Connection failed (" + urlstr + "): " + ioex);
    throw ioex;
}
```

**Any Questions?**

**Secure Authenticated Java  
Communication with JSSE**

RJUG May 11, 2004

Ralph Durkee,  
Security Certs: CISSP, GSEC, GCIH

[www.rd1.net](http://www.rd1.net) [rd@rd1.net](mailto:rd@rd1.net)