CS255: Introduction to Cryptography **Winter 2003**

Programming Project 2

Due: Monday, March 10th 2003, 11:59pm

For programming project 2 you will implement a chat room system with anonymity as well as authentication using SSL (Secure Socket Layer). The project 2 code is completely independent from that of project 1 (except for the GUI part). This project is a bit **bigger** than project 1, so please start early.

For project 2, you will learn

- **keytool (command line utility)** to generate and manage keys and certificates.
- **IAIK (JCE Extension)** to create and sign certificates.
- **JSSE (Java Secure Socket Extension)** to do secure networking.

We will examine each project feature in detail below.

General Description

In this project, you will implement a chat room system which provides anonymity as well as authentication. By anonymity we mean that a client can join the chat room without revealing its real identity to the chat server. By authentication we mean that only clients presenting valid certificates will be allowed to enter the chat room. This can be realized as follows. First, all online parties should obtain long-term certificates from a Registration Certificate Authority (CA). Before contacting the chat server for chat room entrance, each client must obtain a chat-certificate (short-lived certificate) from an online Privacy-CA. Only clients presenting a valid chat-certificate will be allowed to enter the chat room. The chat-certificate anonymizes the client: the subject-name in the chat-certificate is a random pseudonym (a random number). This hat-certificate, issued by the Privacy-CA and sent to the chat-server, should reveal no information about the real identity of the client. The Privacy-CA will issue a chat-certificate only to those clients who present a valid long-term certificate from the Registration-CA.

To prevent misuse, the chat room system has a revocation mechanism. Imagine that a client posts the word “bomb”. The chat server will immediately kick out this client. However, that client could contact the Privacy-CA to obtain a new chat-certificate under using a new random pseudonym and then log back in to the chat server. To prevent this, when revoking chat privileges, the chat-server will contact the Privacy-CA and ask it to stop issuing chat-certificates to the offending client. Consequently, both the chat-server and the Privacy-CA should respectively maintain a CRL (certificate revocation list) and refuse connection requests from clients whose certificates are in the corresponding CRL. Note that the chat-server never learns the real identity of the offending client.

Obtain Long-Term Certificates

In this project, there are four entities: client, chat-server, Registration-CA, and Privacy-CA. Client, chat-server and Privacy-CA are online entities. You may assume that these online entities have the public-key of the Registration-CA, which you can generate once and for all using **keytool**. Communication between online entities in the Chat system should be protected using SSL. We require mutual authentication of both parties establishing a secure connection. Therefore, each party should have a certificate and corresponding private key that it will use to setup the SSL channel. These certificates are generated by the Registration-CA or the Privacy-CA.

The Registration-CA is an offline tool that issues certificates to the online entities. The first thing to do is generate the Registration-CA's private key and self signed certificate using **keytool** and store both in the CA's keystore. The Registration-CA's self signed certificate should be made available to all parties. The CA's private key remains secret (password protected) and is only accessible to the Registration-CA tool.

Next, you will need to use the Registration-CA tool to generate a private-key and a certificate for each of the online entities. Each party's certificate and private key are stored in that party's keystore. Upon startup, the entity (chat-server, client, or Privacy-CA) will read their certificate and private key from their own key store). You need to implement the Registration-CA. The code for generating and signing certificates has been provided for you (See class X509CertificateGenerator).

Obtain Chat Certificates

In this project, the chat server only accepts certificates issued by the Privacy-CA when granting clients the permission to enter the chat room. So each client should first request a chat certificate from the Privacy-CA. The Privacy-CA should issue an ephemeral certificate to a client upon request provided the client is not on its CRL. In the chat certificate, the name of the client should be replaced by a pseudonym (a random number). Hence the server won't know the real identities of clients, and so chat certificates provide anonymity to the chat room system. However, the Privacy-CA should maintain an internal map between pseudonyms and real names to enable certificate revocation (See below).

To obtain chat certificates, the following steps are needed:

- A client establishes a secure connection to the Privacy-CA. Here the client uses its long-term certificate from the Registration-CA.
- The client sends a chat certificate request to the Privacy-CA.
- The Privacy-CA verifies client certificate (issued by the Registration-CA). If the certificate is valid and it is not on the CRL, then the Privacy-CA

creates a random pseudonym, and creates a new certificate where the subject name is this pseudonym.

- The Privacy-CA records the association between the real identity and the pseudonym in a table. It sends the new certificate back to the client.

Revoke Certificates

The chat room system has a censorship policy. Upon receiving a message from a client, the server will check if the message is offensive (You decide what is offensive). If a client violates the policy, the corresponding message won't be posted in the chat room and server will immediately terminate the connection with this client. Moreover the server will add the chat-certificate of this client to its CRL and inform this violation to the Privacy-CA. Correspondingly, the Privacy-CA will identify the real name corresponding to the pseudonym sent by the chat-server, and add the long-term client certificate to its CRL. As a consequence, the Privacy-CA won't issue any chat certificate to the client. Note that the Privacy-CA should only accept revocation requests from the chat-server.

To complete the "certificate revocation" mechanism, the following steps are needed:

- When a policy violation happens the server adds the relevant certificate to its CRL and sends the violator's pseudonym to the Privacy-CA.
- When receiving a revocation request from the server, the Privacy-CA finds the corresponding real name of the client and adds the client's general purpose certificate to its CRL.
- The server should set up a customized TrustManager to initialize SSLContext (See JSSE APIs for details). Upon each SSL handshake, the customized trust manager should check if the certificate presented by the peer client is in its CRL. If so, the server will refuse the connection request from this client.
- The Privacy-CA should also use a customized TrustManager to initialize its SSLContext. Upon each SSL handshake, the customized trust manager should check if the certificate presented by the peer client is in its CRL. If so, the Privacy-CA will refuse the connection request from this client and hence a bad client won't be able to get a chat-certificate.

Implementation

As with the first programming project, we have provided you with starter code. The starter code illustrates the basic socket and thread programming. See the following section for links of tutorials on socket and thread programming. In addition to Sun Java JCE library, you need IAIK JCE extension library to create and sign X509 certificates. The library is in the directory **/usr/class/cs255/lib** and it is also included in the starter code. You will need to modify the ChatClient and ChatServer classes to add the features required for project 2. Here is a description of files we provide for you (files you need to change are in **bold**):

File	Purpose
Makefile	Makefile for the project – modify this to build any classes you add to the project.
Java.policy	Policy file granting network/file permissions.
Chat/ChatLoginPanel.java	GUI class for the login screen.
Chat/ChatRoomPanel.java	GUI class for the chat room screen.
Chat/ChatServer.java	Accept secure connection from clients.
Chat/ChatServerThread.java	Receive messages from clients and post messages to the chat room.
Chat/ChatClient.java	Request chat certificates from the Privacy-CA and send messages to the chat server.
Chat/ChatClientThread.java	Receive posted message from the chat server.
Chat/ClientRecord.java	Store client information.
Chat/ X509CertificateGenerator.java	Generate X509 certificates

You should spend some time getting familiar with the provided framework and reading the comments in the starter code. You will need to copy the /usr/class/cs255/project2 directory to your account. As with project 1, you will also need to source /usr/class/cs255/setup.csh to set your path, classpath and java alias correctly. Building and running the Chat system is much the same as it was for project 1.

Security Libraries and Documentation

In addition to **java.security**.* and **javax.crypto**.*, some classes in **iaik.x509.* and iaik.asn1.structures.*** are also needed to do certificate management**.**

The following are related documentations.

Description	URL
Java1.4 APIs	**http://java.sun.com/j2se/1.4.1/docs/api/**
IAIK JCE Extension API	**http://jce.iaik.tugraz.at/products/01_jce/documentation/javadoc/index.html**
Java **keytool** manual	**http://java.sun.com/j2se/1.3/docs/tooldocs/win32/keytool.html**
JCE Reference Guide	**http://java.sun.com/j2se/1.4/docs/guide/security/jce/JCERefGuide.html**
JSSE Reference Guide	**http://java.sun.com/j2se/1.4/docs/guide/security/jsse/JSSERefGuide.html**
SUN Tutorial of Java Socket Programming	**http://java.sun.com/docs/books/tutorial/networking/sockets/**

SUN Tutorial of Java Thread Programming	**http://java.sun.com/docs/books/tutorial/essential/threads/**
IBM Tutorial of JSSE (Basic)	**http://www-900.ibm.com/developerWorks/cn/education/java/j-jsse/tutorial_eng/j-jsse-5-1.html**
IBM Java Tutorial of JSSE (Advanced)	**http://www-106.ibm.com/developerworks/java/library/j-customssl/?dwzone=java#3**

Some classes you may want to take a look at:

java.security.SecureRandom

java.security.KeyStore
javax.net.ssl.KeyManagerFactory
javax.net.ssl.KeyManager
javax.net.ssl.TrustManagerFactory
javax.net.ssl.TrustManager

java.net.ServerSocket
java.net.Socket
javax.net.ssl.SSLSocket
javax.net.ssl.SSLServerSocket
javax.net.ssl.SSLSocketFactory
javax.net.ssl.SSLContext
javax.net.ssl.SLSessionContext

java.security.cert.Certificate
java.security.cert.X509Certificate
javax.security.auth.x500.X500Principal
iaik.asn1.structures.AlgorithmID
iaik.x509.X509Certificate
iaik.asn1.structures.Name

Help

- The class newsgroup will again be the primary place to look for answers and ask questions.
- At the session on March 4, TA will review APIs for SSL and certificate management. That session only helps if you are familiar with the starter code and the to-do list. We encourage you start as early as possible.
- As a last resort, you can email the staff at cs255ta@cs.stanford.edu.

Submission

In addition to your well-decomposed, well-commented solution to the assignment, you should submit a **README** containing the names, **leland usernames** and **SUIDs** of the people in your group as well as a description of the design choices you made in implementing each of the required features. In particular, you need to point out primary code locations (class names) for each required feature.

When you are ready to submit, make sure you are in your project2 directory and type **/usr/class/cs255/bin/submit**.