Computer Security — Fall 2019
CS 4351 — Professor L. Longpré

Assignment 2

Introduction

The purpose of this assignment is to gain experience in implementing a cryptographic protocol that involves key exchange, digital signatures and encryption.

General description

In this assignment, you are asked to implement a protocol similar to the SSL or the SSH protocols. The protocol involves certificates, key exchange, digital signatures, encryption and decryption. We will use both public key and symmetric encryptions.

Part 1: Creating RSA cryptographic keys

The program `CreatePemRSAKeys.java` creates an RSA key pair, and saves the private and public parts in 2 different files, named `RSAprivateKey.pem` and `RSApublicKey.pem`. The program `CreatePemDSAKeys.java` creates an DSA key pair, and saves the private and public parts in 2 different files, named `DSAprivateKey.pem` and `DSApublicKey.pem`. The files are in PEM format, which is readable in a text editor. It uses the static methods in `PemUtils.java`, which transforms between Java key objects and PEM format. The methods use Base64 encoding. Base64 encoding is included starting in Java 8.
Use your program to create four pairs of keys, two for the client and two for the server. The client and the server will use different key pairs for signature and for encryption. Choose file names that refer to you (like JohnDoeClientSignPublic or JohnDoeServerEncryptPrivate). In this protocol, we will use the RSA algorithm for encryption and the DSA algorithm for signature. Test your keys for encryption and decryption with programs `EncryptRSA.java` and `DecryptRSA.java`. Test your keys for digital signatures with programs `SignDSA.java` and `VerifyDSA.java`.

Part 2: Generating key certificates

I have set up a web server that acts as a certificate authority. The server takes a name, a username, a password, an encryption public key and a signature public key. When submitting the form, it executes a PHP program to produce a certificate. A username (your miners username) and a password was sent to you by email. The name could be any string, like "Joe Client number 5". The username is to identify you as the one being certified, and checking the password is the certification process. The web server uses the RSA algorithm to sign certificates. The server can be accessed at *http://cssrvlab01.utep.educlasses/CS5339/longpre/authority/CertificateRequest.html*.
The authority's public key is provided in file `CA_RSApublicKey.pem`. In practice, the X.509 standard dictates an elaborate structure for digital certificates, and the certificate includes a digital signature. The signature is Base64 encoded to make it readable. For this assignment, to keep it simple, the certificate only contains the name, the date, the username, the encryption public key, the signature public key, and a digital signature of those items with the authority's private key, in a format similar to the PEM format. Include in your report the certificates you generated.

Part 3: Verify your certificates

The program `verifyCert.java` verifies the format and the signature of a certificate. It assumes the certificate is in file `certificate.txt` and that the public key is in `CA_RSApublicKey.pem`. Make sure that it works with the certificates you obtained in Part 2.

Part 4: Communication

The programs `MultiEchoServer.java` and `EchoClient.java` are programs that use sockets to communicate with each other. First run the Server. It waits for connections on port 8008 from clients. Then run the Client. It may be more realistic to use another instance of the IDE instead of running the two programs from the same instance. Better yet, use two different IDEs or run one from the command line. There are many IDEs like Dr.Java, Eclipse, Netbeans. The client asks for an IP address or localhost, which resolves to your computer's IP address. Use localhost when the server and the client

run on the same computer. Also try for something even more realistic: run one program on a virtual machine and another program on your computer. You will need the IP address of the server, which will now be different from localhost. On the Windows command line, use the ipconfig command to get the IP address of your computer. Look for IPv4 or inet address. On Linux, use the ifconfig command. You may have to play with the network settings of your virtual machine. First try both NAT and bridged.
After the client connects, the server echoes all the strings back to the client.