**IS2150/TEL2810 Introduction to Security**

**Programming Project  Report**

**Jing Lin**

In this project, all client and server use localhost and port 7999.

## 1. Crypto Techniques

### A) Authentication

Client is named “ProtectedClient”; server is named “ProtectedServer”.

User name is given as George and the password is abc123.

The output of this program is to show that the “client is logged in”, if it is successfully authenticated.

### B) Signature

Sender is named “ElgamalAlice”; receiver is named “ElGamalBob”.

Alice sends the public key, message and signature to Bob. And Bob will verify. Bob will print out the original message “The quick brown fox jumps over the lazy dog.” and “Signature verified.” if successfully authenticate.

### C) Public-Key System

Sender is named “RSAAlice”; receiver is named “RSABob”.

Both Alice and Bob will generate their own public keys and private keys, and store the public key into “RSAAlicePublicKey.txt” and “RSABobPublicKey.txt” accordingly.

The program will wait till the keys are stored in the files and then require user to press enter to continue. This way will prevent either Alice or Bob access other’s file before keys are properly stored.

Alice then sends the public key, encrypted message, and signature to Bob. Bob will print out the original message “The quick brown fox jumps over the lazy dog.” and “Signature verified.” if successfully authenticate.

### D) X.509 certificates

Client is named “X509Client”; server is named “X509Server”.

To generate the certificate using keytool, enter the following in terminal:

keytool -genkey -dname "CN=Jing Lin, OU=iSchool, O=PITT, L=Pittsburgh, S=Pennsylvania, C=US" -alias jinglin -keypass jinglin -storepass jinglin -keyalg RSA -keysize 1024 -keystore keystore.jks -validity 999

The server sends the certificate to the client. And the client will verify the certificate and print out the certificate as followed. Then the client encrypt and send the message “The quick brown fox jumps over the lazy dog.” The server will print out the decrypted message.

Server piblic key validated

Certificate:

[

[

Version: V3

Subject: CN=Jing Lin, OU=iSchool, O=PITT, L=Pittsburgh, ST=Pennsylvania, C=US

Signature Algorithm: SHA256withRSA, OID = 1.2.840.113549.1.1.11

Key: Sun RSA public key, 1024 bits

modulus: 107897872740636894157847974045574016246986538057072835722754527327016854130078696307983184118673756330575232139301438777198307949851305972685717648061181862958033412231438254736513737558329579913854692263740504809880690960012190924416590750645327887264500785944793741282352468576918468327292270524750878084217

public exponent: 65537

Validity: [From: Sat Nov 08 16:18:29 EST 2014,

To: Fri Feb 06 16:18:29 EST 2015]

Issuer: CN=Jing Lin, OU=iSchool, O=PITT, L=Pittsburgh, ST=Pennsylvania, C=US

SerialNumber: [ 4eb7a935]

Certificate Extensions: 1

[1]: ObjectId: 2.5.29.14 Criticality=false

SubjectKeyIdentifier [

KeyIdentifier [

0000: CE 3A 5A 99 B0 BC 34 04 91 B6 50 3D 6C 8E 6B A2 .:Z...4...P=l.k.

0010: 25 3D 99 A8 %=..

]

]

]

Algorithm: [SHA256withRSA]

Signature:

0000: 51 BA F9 1B 5D F1 8A 53 D8 C7 21 31 47 7F F2 63 Q...]..S..!1G..c

0010: 45 8F 86 70 D2 18 7D 1B C2 B1 A5 3D 52 86 0E B8 E..p.......=R...

0020: B8 75 92 CF A6 1D B7 A0 4E DA 79 66 19 5A DD A8 .u......N.yf.Z..

0030: 56 67 E5 D6 47 81 F3 E8 EC 91 E5 5A FD 9D E7 7B Vg..G......Z....

0040: 25 E6 93 CE 12 B1 C2 41 85 1C 40 7F 39 3E B9 DF %......A..@.9>..

0050: A7 96 11 D9 DE 38 50 B8 03 4D 5C 4B 07 C5 F0 48 .....8P..M\K...H

0060: C6 25 27 EF 98 8B A6 AC 4C DA 6D 35 79 BB 4E B5 .%'.....L.m5y.N.

0070: 45 CA 7B CD C2 B6 41 18 51 D1 B0 2D 95 B5 78 93 E.....A.Q..-..x.

]

The quick brown fox jumps over the lazy dog.

### Answers to the question:

What are the limitations of using self-signed certificates?

Self-signed certificates require both the client and server to interact (server to send the certificates and client to accept and verify), while comparing to certificate-authorities-signed certificates, the client only needs to get the certificate from authorities.

Also, it is relatively hard to deal with large number of clients, and difficult to maintain as every time the certificates is changed, it has to be sent to the all the communicating partners again.

What are they useful for?

It is efficient when the server has small number of clients.

Self-signed certificates are relatively easy to generate and customize, the server can change its key size, validity duration, etc..

It is generally free, while CA certificates require fees.

## 2. Password Cracking

### Part 1: Implementation of password authentication mechanism

The first program is named “PasswordCracking1\_Register”. It will prompt “Please enter user name” and “Please enter password” and allow user to enter accordingly. The program will then store the user name and the hash code of password into file named “password.txt”.

The second program is named “PasswordCradking1\_Login”. It will prompt “Please enter user name” and “Please enter password” and allow user to enter accordingly. And it will print “password matched” if user is found and successfully authenticate; it will print “password not match” if user name is found but password not match; it will print “user not found” if user name is not match.

### Part 2: password cracker

The program is named “PasswordCracker2”. It will prompt “Please enter user name” to allow entering the user name that you want to crack the password for. Then the program will print out the user’s password’s hash code get from the file “password.txt”.

The program will use type 1 cracker first to try crack the password. If the password is found, it will print out the password and the time it takes to crack it.

Then the program will use type 2 cracker. It first tries generating words that added 1 special character (k=1) based on words from the “dictionary.txt”, and then 2 special characters (k=2) until it finds the password. The program will also print out the progress. It will printout the password and the time takes to crack it.

Example output:

Example 1:

Please enter user name

aaa

passwordHash: 263addb7611158a2c3db7c164a20cc2f

Password found:lamp

type 1 time taken: 464

k=0

0% 11% 22% 33% 44% Password found:lamp

type 2 time taken: 737

Example 2:

Please enter user name

zzz

passwordHash: 884b79e1a35244c663815e9c921a451d

could not find password with type1

type 1 time taken: 550

k=0

0% 11% 22% 33% 44% 55% 66% 77% 88% 99%

k=1

0% 11% 22% 33% 44% 55% 66% 77% 88% 99%

k=2

0% Password found:$appeal#

type 2 time taken: 64171