CSE 543 - Fall 2018 - Project 1: SSH File Transfer

1 Dates

- **Out:** *September 11, 2018*
- **Due:** *October 3, 2018*

2 Introduction

In this project, you will develop a client-server system that provides secure file transfer. You will implement the SSH protocol to construct a secure channel between the client and server over which to perform secure file transfer. You will use the OpenSSL library to implement the SSH protocol. The SSH protocol produces a symmetric key shared between the client and server, and you will use the OpenSSL library once again to use that key to transfer the file.

The system you produce must run on the Westgate Linux lab machines. These machines are named `cse-p204instXX.cse.psu.edu`, where `XX` is a number between 01 and about 40. All these machines should be identical and already have the OpenSSL library installed.

3 Overview

The secure file transfer system will work as follows.

The server program will be started by the following command line: `cse543-p1-server <private-key-file> <public-key-file>`: where (1) the `<private-key-file>` is the name of the file that stores the private key for the server and (2) `<public-key-file>` is the name of the file that stores the corresponding public key for the server.

Please create the public and private keys using the OpenSSL system using the following commands. Please create your public key in the RSA format for using the provided RSA functions for OpenSSL.

```
# generate key pair - mykey.pem holds private key
openssl genrsa -out mykey.pem 2048
# extract public key in basic format - pubkey.pem is in PKCS#8 format
openssl rsa -in mykey.pem -pubout -out pubkey.pem
# convert public key to RSA format - rsapub.pem holds public key
openssl rsa -pubin -in pubkey.pem -RSAPublicKey_out > rsapub.pem
```

The client program will be started by the following command line: `cse543-p1 <file-to-transfer> <server-ip-address> 1 1`: where (1) the `<file-to-transfer>` is the file pathname of the file to transfer from the client to the server and (2) `<server-ip-address>` is the IP address of the server host. The last two arguments are fixed - we'll likely use them later.

Start the server first, as it will wait for connection requests from clients. When a connection request is received from a client the sequence of steps will be performed.

Perform the SSH protocol: The client will initiate the SSH protocol to produce a symmetric key to be shared by the client and server.

Transfer the file: The `<file-to-transfer>` will be sent encrypted and integrity protected from the client to the server. The server will store the file in a directory called "shared" under the directory from which the server is run.

Server awaits next request: The client will terminate and the server will await the next request from the next client.

4 Project Tasks

The initial version of the program includes two functions `test_rsa` and `test_aes` that encrypt a message from an old cartoon that I barely remember. Really old.

These functions demonstrate how to perform symmetric and public key encryption with OpenSSL library, which should be a big help in the project.

I suggest you perform the project tasks listed below in the following order.

1. **Download project tarball**:

   From `http://www.cse.psu.edu/~trj1/cse543-f18/p1-assign.tgz`. The project includes source code and a Makefile for building the tarball for your project for submission.

2. **Write the functions to build encrypted messages for sending and decrypted received messages**: There are two pairs of functions for you to implement: `seal/unseal_symmetric_key` for public key crypto and `encrypt/decrypt_message`. These encryption functions must perform encryption and produce buffers containing the data necessary for the other party to decrypt. The decryption functions must extract the necessary information from a sent buffer and perform the decryption.

3. **Write the function to generate pseudorandom values**: Develop the function `generate_pseudorandom_bytes` by using the OpenSSL functions for producing pseudo-random values.

4. **Develop the SSH Protocol**: Implement the client and server portions of the SSH protocol, as described in the paper for 10/16. There are two functions `client_authenticate` and `server_protocol` to be implemented.

5. **Tranfer the file securely**: Implement the encryption and transfer functionality to send the file from the client to the server in the function `tranfer_file`.

4.1 Download Tarball

The tarball consists of 5 C files and associated header files (for all but `cse543-p1.c` and a Makefile for compiling the program code and producing tarballs.

The C file `cse543-p1.c` is the main file in the project and the file that starts the server and client, depending on which is being built.

The C file `cse543-proto.c` implements the SSH protocol and secure file transfer. All of the code you need to write is in this file.

The C file `cse543-ssl.c` contains the code for leveraging the OpenSSL API. I have downloaded most of this code from the internet, and included the URLs of the sites, so you can read the associated text for these operations. There are some questions on using the OpenSSL API below.

The C file `cse543-network.c` implements network functionality for the client and server.

The C file `cse543-util.c` provides some utility functions used by the implementation.

4.2 Build and Decrypt Encrypted Messages

Start by making sure that you can generate encrypted messages that you can send and decrypt on the other end for both public key and symmetric key crypto. `test_rsa` and `test_aes` show how to generate encrypted data and deecrypt that, but you need to be able to send the encrypted data to the other process in a manner that enables decryption.

4.3 Generate Pseudorandom Values

OpenSSL has a set of functions for generating pseudorandom data. Use these functions to generate pseudorandom data of requested byte sizes.

4.4 Implement SSH Protocol

The main task in the project is to implement the SSH protocol as described in the paper. Since any user is authorized to upload a file in this project, you only have to implement Steps 1-4 of the protocol, corresponding to messages 1-4 in the `ProtoMessageType` enumeration in `cse543-proto.h`. Use the OpenSSL functions provided to implement the SSH protocol.

Note that you only have to supply one server public key in Step 2, which simplifies things a bit. Otherwise, you implementation should achieve the same effect.

Please write the SSH protocol as a series of cryptographic messages in using the course's crypto notation.

4.5 Transfer Files Securely

Once the SSH symmetric key has been shared, the client may transfer its file (`file-to-transfer`) to the server. Only one file will be tranferred per use (i.e., no need to support wildcards, directory copies, etc.).

You need to extend the function `transfer_file` to collect the next file block, perform encryption of the file data in the block, generate the message to send, and send the encrypted file block message to the server. The function `receive_file` (for the server) is provided, which may help a bit.

See the other messages in `ProtoMessageType` enumeration in `cse543-proto.h` for choosing those for file transfer.

5 Testing

I will test your submission on machines in the Linux lab in W204 Westgate. The machines are named cse-p204instXX.cse.psu.edu, where XX is a number from 01 to at least 40.

You should SSH into those machines to verify that your code works. I developed and tested the project code on those machines, so should work fine, but it is up to you to make sure.

You will need to speak to the CSE IT folks if you do not have access to those machines.

6 Deliverables

Please submit the following:

1. A tarball of your password produced with the provided Makefile using `make tar`.
2. PDF file containing the SSH protocol specified using the course's crypto notation.

7 Grading

The assignment is worth 100 points total broken down as follows.

1. I can build and run what you have submitted without incident (10 points).
2. Encrypted communication works (10 points).
3. Generate pseudorandom data (10 points).
4. SSH protocol implementation and specification (60 points)
5. Transfer file securely (10 points).