Computer Security
Programming Assignment 3

Due: 1st May, 2018

Overview

The goal of this project is to introduce you to the real-world applications of cryptography.It will involve the creation of a digital certificate as well as the use of both symmetric and asymmetric ciphers.

Your programs should validate in **valgrind** and be free of memory errors. Use of C or C++ is required, and the program must compile and run in the CS department labs. You code should also be **modularized** and **well-documented**(commented).

Part I

You will need to obtain a signed certificate from a certificate authority (e.g., CAcert) in this step. The CA will issue you a PKCS #12 key package, which will contain your public key, your private key, and the CAs signature. If you do not want to obtain a certificate from a CA, you may also generate your own self-signed certificate. See the OpenSSL documentation or other online resources for details.

Then, you will need to use OpenSSLs command-line utilities to extract the public and private keys and store them in the PEM format (e.g., as **pubkey.pem** and **privkey.pem**). This is a Base-64 encoded format which you will be able to view in a standard text editor.

Part II

Now, you must create a C or C++ program which is capable of using your public key, a third party public key, and an encrypted session key to create a symmetric-key ciphertext and an asymmetric signature. Your plaintext file should include at least the names and Banner IDs of your group members; you may also include any other information as you see fit.

You are required to use your DES implementation from Programming Assignment 2 for the symmetric cipher. If you did not complete your DES implementation, you may use the OpenSSL library functions (*which will incur a 10-point penalty*) or use the **system** function to call the OpenSSL command-line utility (*which will incur a 20-point penalty*).

The OpenSSL EVP library will be necessary for performing asymmetric (RSA) cryptography.It provides facilities for hashing, signing, encryption, etc.

Succinctly, your program must do the following:

1. Take the filenames of the plaintext message, the encrypted session key, the third-party public key, and your private key as command-line parameters.

2. Use the third-party public key to decrypt the session key.
3. Save the plaintext session key to a text file.
4. Use the DES session key to encrypt the plaintext.
5. Use your private key to sign the encrypted message.
6. Save the ciphertext and signature to an output file (or separate output files).

Part III

Now, having created an encrypted and signed message, you must create another program which is capable of decrypting it and verifying the signature.

This program must take your public key, the plaintext session key, the ciphertext file, and the signature file (if it is separate) as parameters. It should decrypt the ciphertext and print theresult, as well as state whether or not the signature was determined to be authentic.

Again, you should use your own implementation of DES, and use OpenSSLs EVP library to perform RSA and signature validation.

Submission instructions Submit a tarball (**.tar** or **.tgz**) containing the following items:

- Your public key, in **.pem** format.
- The signed ciphertext you produced.
- Your C or C++ files, with a **Makefile** which compiles them.
- A **Readme** which explains how to use your programs.

Make it clear which program is for Part II and which is for Part III; you may name them **encryption** and **decryption**, or **part2** and **part3**; any naming scheme is fine as long as it isclear which is used for which task.

Remember that your code must be modular, must be documented, must validate in **valgrind**s **memcheck** tool, and must compile and run successfully in the CS labs.

Resources

- CAcert, a free certificate authority:
  `http://www.cacert.org/`
- OpenSSL EVP manual:
  `https://www.openssl.org/docs/manmaster/man7/evp.html`
- Valgrind Memcheck manual:
  `http://valgrind.org/docs/manual/mc-manual.html`