CSCI5470 (Spring 2013)

Assignment 1: My Crypto Utility
Full marks: 100% + 5% Bonus

Due on 21st February 2013, 23:59:59

1 Introduction

Many students are storing their homework assignments on the cloud. One popular tool is Dropbox. Dropbox provides security protection for your data, such that the data will be encrypted before being relayed to the cloud (which is Amazon S3). However, you may not trust Dropbox at all, since it will see all the content of your data. Instead, you may want to encrypt your data *locally* before you send the data to Dropbox.

In this assignment, you will use OpenSSL to develop a utility called *MyCrypt* that applies security protection to a file or a group of files locally. Our security goals are to provide guarantees of confidentiality, integrity, and authenticity.

2 Protocol Description

There are many ways to achieve our desired security goals. We pick the following particular implementation for our grading purpose.

Preliminaries. Before we use MyCrypt, we need to have two sets of public/private key pairs. One is the *long-term key pair*, and another is the *short-term key pair*. The long-term public key is included in a X.509 certificate that is signed by a certificate authority. All keys are stored in separate files in PEM format, while the private key files are protected with a passphrase. We assume that the long-term public/private key pair is based on 1024-bit DSA.

Encryption operation. Given a plain file F, MyCrypt will generate an encrypted file with the format as shown in Figure 1.

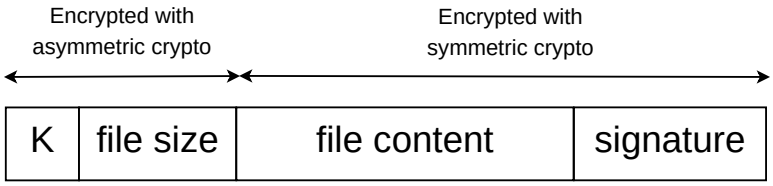

Figure 1: Encrypted format.

The file content is the content of file F. The signature applies to the file content of F based on 1024-bit DSA, using our long-term private key. The file size stores the actual size of the file, which we assume is a 4-byte unsigned integer (in big endian). The file content and the signature are encrypted with a key K via symmetric encryption, while K and the file size will be encrypted with asymmetric encryption (see below). Here, we choose 256-bit AES with CBC. Note that the file content may not be of multiples

of the AES block size, so you need to pad the file content with zeroes. This explains why you need to store the file size[1].

How should we generate K? Here, we apply a double-hash to the file size and the file content with SHA-256, defined by H(H(size, content)). The resulting 256-bit value will be used as K. Also, we will use 0 as the initialization vector for the encryption.

The key K and file size will be encrypted with asymmetric encryption, using our short-term public key. We assume that the short-term public/private key pair is generated by 1024-bit RSA. We use the padding scheme RSA_PKCS1_OAEP_PADDING. You need to think about how to make K and file size into blocks that can be encrypted with RSA using this padding scheme.

All encrypted content will be stored in a file with the same filename, but with the additional extension `.enc` (e.g., we encrypt file `foo.c` and store it as `foo.c.enc`).

Decryption operation. Given an encrypted file, you can restore the original file, as long as you have the long-term/short-term private keys and the corresponding passphrases.

Implementation. The MyCrypt utility handles the encryption/decryption operations. It can apply cryptographic operations on per-file basis.

The options are described as follows.

- `-e`. The encryption operation is performed.
- `-d`. The decryption operation is performed.
- `-f filename`. The file to be encrypted/decrypted.
- `-cert cert.pem`. The certificate that includes the long-term public key.
- `-lpri lpri.pem`. The long-term private key.
- `-spub spub.pem`. The short-term public key.
- `-spri spri.pem`. The short-term private key.
- `-lp passphrase`. The passphrase to access the long-term private key file.
- `-sp passphrase`. The passphrase to access the short-term private key file.

Examples:

- Encryption on file `file.pdf`:
  `./mycrypt -e -f file.pdf -lpri lpri.pem -spub spub.pem -lp 5470`.
- Decryption on file `file.pdf.enc`:
  `./mycrypt -d -f file.pdf.enc -cert cert.pem -spri spri.pem -sp 0745`.

Hints. You may find the function `stat()` useful.

3 Milestones

MyCrypt should achieve the following functions:

- Correct encryption (50%)

[1] OpenSSL provides the EVP library that takes care of all the padding function. Basically, the EVP APIs are the wrapper functions of the low-level cryptographic functions. For learning purposes, we do not use the EVP library except for the certificate-related functions. See class notes.

- Correct decryption (50%)
- **Bonus (5%):** Extend MyCrypt to support batch encryption/decryption. That is, given a directory name, you encrypt/decrypt each of the individual files inside the directory and all its subdirectories.

4 Submission Guidelines

You *must* at least submit the following files, though you may submit additional files that are needed:

- *mycrypt.h*: the header file
- *mycrypt.c*: the mycrypt program
- *Makefile*: a makefile to compile the mycrypt program.

Your programs must be implemented in C. They must be compilable using `gcc` on the Linux machines with OpenSSL installed. Please make sure that the programs can be executed correctly inside the VMs (the TAs will explain the details during tutorials regarding how to test your assignments). Also, your programs should be compiled without any warning message. Use `gcc -Wall` to see if any warning message exists. Any program that is not compilable nor free of warning message will receive zero marks!!

Please refer to the course website for the submission instructions.

The assignment is due on 21st February, 2013, 23:59:59.

Have fun! :)