Lab 1: Basic Cryptography - AES and RSA

January 10, 2019

1 Overview

In this lab, you will learn to use AES and RSA for secure network communications. These two cryptographic algorithms are used in our daily life (e.g. HTTPS protocol). You will need to analyze the performance of OpenSSL implementations of both algorithms, and then you will use them to set up secure communication between yourself (client) and our server to obtain a secret.

2 Using OpenSSL Library

OpenSSL is a software library for applications that secure communications over computer networks against eavesdropping or need to identify the party at the other end. It is widely used in Internet web servers, serving a majority of all web sites. You can download the library from https://github.com/openssl/openssl.git. You will need to checkout the repository and compile it on your machine.
To checkout the repository:

```
$ git clone https://github.com/openssl/openssl.git
```

To compile it on your machine:

```
$ cd openssl
$ ./config
$ make
```

Once it is compiled, you will see a static library *libcrypto.a* in the openssl directory. This library will be statically linked against if you are going to use its RSA and AES implementations.

3 Lab module 1: Performance of Ciphers

The OpenSSL library has the following APIs for RSA

1. RSA_public_encrypt
2. RSA_public_decrypt
3. RSA_private_decrypt
4. RSA_private_encrypt

and for AES

1. AES_encrypt
2. AES_decrypt

First understand how to use these ciphers, and write your programs to measure the performance of *RSA_public_encrypt* and *AES_encrypt* functions. Use and time these functions to encrypt a 16-byte data. Collect one million timing samples for each function. You will need to look up appropriate timmer and also the parameters for AES_encrypt() and RSA_public_encrypt().

The pseudo code for measuring AES is:

```
for (i = 0; i < 1000000; i++){
  t1 = timer_start()
  AES_encrypt()
  t2 = timer_stop()
  record(t2 - t1)
}
```

The pseudo code for measuring RSA public key implementation is:

```
for (i = 0; i < 1000000; i++){
  t1 = timer_start()
  RSA_public_encrypt()
  t2 = timer_stop()
  record(t2 - t1)
}
```

To allow your programs to use these cryptographic functions included in the OpenSSL library, you can refer to the following Makefile for header files inclusion and static linbary linking. Note to update **OPENSSL** setting to the directory of your downloaded OpenSSL:

```
CC=gcc
OPENSSL=../../openssl
INCLUDE=$(OPENSSL)/include/
CFLAGS=-c -I$(INCLUDE)

all: program

program: program.c
  $(CC) program.c -I$(INCLUDE) -o program $(OPENSSL)/libcrypto.a

clean:
  rm -rf program
```

Plot timing distributions of these samples and find the mean for each function.

Question 1: First compare the performance of RSA and AES (on the same size of plaintext - 16 byte), how much slower is RSA than AES? With this implementation cost, what scenaria is RSA mainly used in? What is AES mainly used for?

4 Lab module 2: AES Encryption Mode

There are several operation mode of AES. In this section, you will explore two of them: ECB and CBC. Using each of these two modes to encrypt an image file and report your finding.

To simplify the experiment, we will be using an image in the PPM format. First, separate the header and body sections of the image:

```
$ head -n 4 penguin.PPM > header.txt
$ tail -n +5 penguin.PPM > body.bin
```

Encrypt the body sections and reconstruct the image using the header and encrypted body sections.

```
$ openssl enc -aes-128-ecb -nosalt -pass pass:"A" -in body.bin -out enc_body.bin
$ cat header.txt enc_body.bin > ecb_penguin.ppm
```

Question 2: Compare these two encrypted images, and comment on the security. What is downside of CBC mode in terms of performance? Suggest one operation mode you think is the best and give your reason.

5 Lab module 3: Secure Communication

You will need to use both AES and RSA to communicate with a service we hosted at *ecehss.coe.neu.edu* at port *12000* and obtain a 16-byte secret. The following is the communication protocol:

```
server -> [size of server's public key] -> client
server -> [server's public key] -> client
server <- [encrypted size of the client's AES key using server's public key] <- client
server <- [encrypted client's AES key using server's public key] <- client
server -> [encrypted secret using client's AES key] -> client
connection close
```

You can find a good tutorial about socket programming at https://www.geeksforgeeks.org/socket-programming-cc/

6 What You Need to Turn In

For the lab submission, you will need to turn in all your code, which should be testable by us using Makefile commands, and the following writeup items:

1. Figures showing the timing distribution for AES and RSA functions. For AES, name the figure file as *aes.pdf*, and for RSA, name the figure file as *rsa.pdf*. Your Makefile should contain a target *p1*. When we run the command *make p1*, it should generate *aes.txt* and *rsa.txt*. Each text file should contain timing samples you used to generate your submitted figures.
2. Answers to **Question 1** and **Question 2**, and save your answers in *answers.txt*
3. The secret; save the secret in *secret.txt*. Your Makefile should contain a target *p3*. When we run the command *make p3*, it should print out the *secret*.

For your programs, include a Makefile and a README file for how to compile and run your programs. Use the following command to zip your submission.

```
$ zip [your last name]-lab1.zip rsa.pdf aes.pdf answers.txt secret.txt codes
```

7 Resources

Your best friend when working with security is Google. The following resource links may be useful.

- Getting started with Linux Commands: http://www.usm.uni-muenchen.de/people/puls/lessons/intro_general/Linux/Linux_for_beginners.pdf
- Getting started with C programming: http://www.kciti.edu/wp-content/uploads/2017/07/cprogramming_tutorial.pdf (see page 6)
- Getting started with Makefile: http://mrbook.org/blog/tutorials/make/

- Getting started with Matlab programming: http://mayankagr.in/images/matlab_tutorial.pdf
- Getting started with Python programming: https://www.cs.uky.edu/ keen/115/Haltermanpythonbook.pdf