CSCE4013/5013 Homework 4 (Programming)

Due date: October 19, 2018

Full Grade: 100 pts

I. Task Description

In this assignment, you will implement authenticated communications between two parties, Alice and Bob, and evaluate the performance of HMAC and RSA digital signature. For simplicity, Alice and Bob will be simulated by two programs running on the same computer. When Alice sends a message to Bob, she writes the message to a file. Bob receives the message through reading from the file.

Part 1: Implement authentication/integrity protection using HMAC with SHA-256 as the underlying hash algorithm. Assume that Alice and Bob already have a 16-byte shared secret key *k* (e.g., they can read the key from the same file). Alice generates the HMAC of an 18-byte message *m* (the message is manually input from command line) using key *k*, and writes the message m as well as the HMAC into a file named *mactext*. Bob reads the message and HMAC from the file, verifies the HMAC, and prints whether the verification succeeds.

Part 2: Implement digital signature using RSA with 2048-bit key. Assume that Bot already has got Alice's public key (you need to figure out a way to do this). Alice signs an 18-byte message *m* (the message is manually input from command line) using her private key to get the signature *s*, and writes the message *m* as well as the signature *s* into a file named *sigtext*. Bob reads the message and signature from the file, verifies the signature using Alice's public key, and prints whether the verification succeeds.

Part 3: Measure the performance of HMAC and digital signature. Take a 7-byte message *m* manually input from command line. Implement HMAC with SHA-256 as the underlying hash algorithm. Generate the HMAC of the 7-byte message using a 16-byte key for one hundred times, and measure the average time needed for one HMAC generation. Implement RSA with 2048-bit keys. Generate the digital signature of the 7-byte message and verify it for one hundred times, measure the average time needed for one signature generation, and the average time needed for one signature verification. Print the average time of HMAC generation, signature generation, and signature verification separately.

Part 4: Hash collision and birthday paradox. Suppose Dr. Smart has designed a hash function H() that uses the first 8 bits of the SHA-256 hash value of a message as the message's hash; i.e., H(m)={The first 8 bits of SHA-256(m)}. **4(a)** Generate random messages and compute their hash values using H() until you find two different messages that have the same hash value. Record the number of messages you tried to find the collision, and print the two messages that have the same hash value as well as their hash value. **4(b)** Repeat this process for 20 iterations and calculate the average number of trials needed to find a collision. Print out the average number of trials needed to find a collision. (For the 20 iterations, don't print the number of trials, messages and hash value for each iteration to have a clean screen. Only print out the final average number of trials needed to find a collision.) **4(c) Optional (do this offline and do NOT submit this sub-part)** Try changing 8 to 16, i.e., H(m)={The first 16 bits of SHA-256(m)}, and see the average number of trials needed to find a collision..

II. Tests

You need to demo your program to the instructor. A demo sign-up sheet will be distributed in class later. During the demo, your program will be tested in the following ways.

Test of Part 1: Your program needs to take a manually input plaintext message from the command line, print the derived HMAC, and print whether the verification succeeds. Then a manual change will be made to the *mactext* file (either to the message m or to the HMAC), and print whether the verification succeeds.

Test of Part 2: Similarly, your program needs to take a manually input plaintext message from the command line, print the derived RSA signature, and print whether the verification succeeds. Then a manual change will be made to the *sigtext* file (either to the message m or to the signature), and print whether the verification succeeds.

Test of Part 3: Your program needs to take a manually input plaintext message from the command line, and print the average time of HMAC generation, signature generation, and signature verification separately.

Test of Part 4: For 4(a), your program needs to print the two messages that have the same hash value as well as their hash value. For 4(b), your program needs to print out the average number of trials needed to find a collision. 4(c) will not be demoed.

III. Other Instructions

Any programming language is fine.

Messages must be manually input from command line (not from a file). Those required to be "Printed" must be printed to the command line (not to a file).

Submit your source code, your **executable**, and necessary support files (e.g., an empty *mactext* file, *sigtext* file, and, if any, the file(s) used to exchange shared secret key and public key) to Blackboard as a .zip file named in this format: HW4.YourLastName.YourFirstName.zip.