ECE 443/518 Fall 2022 - Project 1

The Password

Report Due: 10/2 (Sun.), by the end of the day (Chicago time)
Late submissions will NOT be graded

I. Overview

Prof. Wang sent out an encrypted message to ECE 443 students. Unfortunately he forgot the password as well as the original message. Prof. Wang still remembers how he obtains the encrypted message: first, a 256-bit key is generated as the SHA256 sum of a 4-digit password; then, AES-GCM is used to encrypt and authenticate the message with the nonce including all 0 bytes and the additional data being his email address.

For your conveniece, Prof. Wang has put what he remembers in a Go program here. Now it is your turn to find the password and the original message using knowledge of cryptographic hash functions and ciphers.

II. Cryptographic Hash Functions and Ciphers

Before you actually write code to find the password, you will first need to understand how to compute SHA256 hash and encrypt/decrypt with AES-GCM properly in Go. You will need to read the two functions validateSHA256() and validateAESGCM() and to observe their outputs in order to understand how they work. Answer the following 4 questions in your project report:

1. From the output of validateSHA256(), decide the length of the hash generated by SHA256. Is it as expected?
2. What is the length of the nonce used for AES-GCM?
3. What is the length of the plaintext in validateAESGCM()?
4. What is the length of the byte slice returned by the Seal function? Why is it named ciphermac?

III. Find the Password

Now you are ready to modify the function findPassword() to find the correct password and then the original message. I have provided a loop to generate all possible 4-digit password and you will need to add code to locate the correct one via brute-force attack.

IV. Bonus: Performance Evaluation

Finding the correct password among 4-digit strings for Project 1 won't require much computational power. However, if there are more choices, we need to evaluate how many we can check within a reasonable amount of time. For a 20% bonus, please evaluate the performance of SHA256 and AES-GCM using the following settings.

- Time the process to compute the SHA256 hash of a 16-byte message.
- Time the process to encrypt a 1M-byte message (1024*1024 bytes) with AES-GCM using 256-bit AES key and no additional data.
- Time the process to decrypt the message above with AES-GCM using 256-bit AES key and no additional data.

Note that you will need to do your own research to find how to measure time in Go, and you may need to run the experiment for many times in order to obtain meaningful numbers.

IV. Deliverables

Submit the following to Blackboard for this project.

1. 30 Points: Your source code that can be executed correctly.
2. A project report including the items below.
    - 20 Points: answers to the four questions in Section II.
    - 50 Points: explain your approach and results for Section III.
    - 20 Points (bonus): explain your approach and results for Section IV. Do the performance measurements meet your expectations?

The project should be done individually. You can discuss the project with other students but all the source code and writings should be your OWN. PLAGIARISM and called for DISCIPLINARY ACTION. NEVER share your source code and reports with others.