Exercises for Module 4

The exercises for this module are about hash functions, MACs and public-key cryptography. Live-coding scripts and selected solutions will be posted on the Github repository for the course: https://github.com/henningth/Applied-Cryptography-2024

Hash functions

Exercise 1: Starting from the first live-coding (about MD5 and SHA1), add functionality so that the user can enter a string, which the program then computes and displays the MD5 and SHA1 hashes of. Test with strings that are similar, such as “Hello” and “H3llo”.

Exercise 2: Starting from the first live-coding (about MD5 and SHA1), compute the MD5 and SHA1 hash of a file or your choosing. Have the program print the filename as well as the MD5 and SHA1 hashes of it. (Hint: To open a file in the same directory as the Python script, you can use:

filepath = os.path.dirname(__file__) + "\\" + filename

where filename is the name of the file.)

Exercise 3: Continuing from the previous exercise, download the two PDF files from https://shattered.io. Compute the SHA1 hash of each of the two files, what do you observe? What about the MD5 hashes of them?

Exercise 4: This exercise is about requirements of cryptographic hash functions:

(a): Why do we want a hash function to be one-way?

(b): Suppose we have a hash function which takes a 128-bit input and gives us a 64-bit output. Must it be the case that there is at least one colliding pair of inputs for this hash function?

Message Authentication Codes

Exercise 5: Consider the following method for generating a tag for a message: Given data, the tag is the SHA256 hash of the data. Is this method secure, or are there some problems here?

Exercise 6: Starting with the live-coding with the MAC, implement a simple Encrypt-then-MAC scheme. Be sure to use two different keys for the Encryption and MAC respectively. Test your scheme on some plaintexts and print the resulting ciphertexts and associated MAC tags. Use AES-CTR for encryption and HMAC-SHA256 for the MAC.

Exercise 7: This exercise is a continuation of the previous one. Implement a simple Decrypt-then-MAC scheme. Test you scheme on some (ciphertext, tag)-pairs that you generated in the previous exercise.