Exercises for Module 5

The exercises for this module are about public-key cryptograpy (RSA), TLS, and the Diffie-Hellman key-exchange. Live-coding scripts and selected solutions will be posted on the Github repository for the course: https://github.com/henningth/Applied-Cryptography-2024

RSA

Exercise 1: Consider RSA, where your private key is d = 937 and your public key is (n, e) = (2537, 13). You receive the ciphertext c = 2222.

(a): What is the modulus in this setup?

(b): Decrypt the ciphertext you received, what do you get?

(c): Encrypt the plaintext obtained in part (b). What do you observe?

Exercise 2: This exercise is about malleability of RSA. We use the private key d = 937 and public key (n, e) = (2537, 13). Note that these are the same as in the previous exercise.

(a): Choose two different plaintexts (i.e. integers) that you want to encrypt. Encrypt these using the given RSA parameters.

(b): Multiply the two ciphertexts and then decrypt the result of the multiplication. What do you observe in the decryption of the recovered plaintext with respect to the two plaintexts in part (a)?

Exercise 3: In this exercise, you are to encrypt and decrypt a message using RSA in the Cryptography library. Start by downloading today's RSA starter-code from GitHub.

(a): Run the code which generates RSA keys and saves them in separate PEM files. You should now have two PEM files, one for the private key and one for the public key. Use the parameter 65537 as public exponent, and a key size of 2048 when generating the keys.

(b): Encrypt a message of you own choice using RSA, using the code you downloaded. Use the public key for encrypting your message.

(c): Decrypt the message you encrypted in the previous part. Which key do you need to use?

(d): Encrypt the file Dice.png using RSA. Does it work? If not, what is the problem?

(e): Check what is the largest number of bytes you can encrypt using RSA with the specified parameters. (Hint: it is sufficient to test with random plaintexts)

(f): Based on the above, do you think that RSA is suitable for encrypting large files? If not, suggest an alternative approach.

Diffie-Hellman

Exercise 4: Write a Python program which implements the anonymous Diffie-Hellman key exchange. Use p=23 and g=5 as parameters. For this part, it is sufficient to test the key exchange in the same script.

Exercise 5: Continuing with Diffie-Hellman, using the same parameters as in the previous exercise, write a Python program that implements a Man-in-the-Middle attack (see the slides for hints).

TLS

Exercise 6: Why are nonces used in the handshake in TLS?

Exercise 7: Suppose Alice and Bob communicate over TLS. How can Alice be sure that she is in fact communicating with Bob?

Exercise 8: How does TLS prevent MITM attacks?

Exercise 9: Suppose Bob initiates a TCP connection with Eve who is pretending to be Alice. During the handshake, Eve sends Alice's certificate to Bob. In what step of the handshake algorithm will Bob discover that he's not communicating with Alice?

Exercise 10: What is the purpose of the Preshared Master Secret?

Exercise 11: Why is the symmetric encryption key used when Alice sends data to Bob different from the symmetric encryption key used when Bob sends data to Alice?

Exercise 12: What is the purpose of using a MAC in TLS?

Exercise 13: Suppose Alice and Bob are communicating over an TLS session. Suppose an attacker, who does not have any of the shared keys, inserts a bogus TCP segment into a packet stream with correct TCP checksum and sequence numbers (and correct IP addresses and port numbers). Will SSL at the receiving side accept the bogus packet and pass the payload to the receiving application? Why or why not?

If you've solved the above exercises, you can look at exercises from previous lectures that you didn't solve.