Exercises for Module 2

The exercises for the first module are about exclusive OR and block ciphers. Live coding and solutions to some of the exercises will be posted on the Github repository for the course: https://github.com/henningth/Applied-Cryptography-2024

Exclusive OR repetition

Exercise 1: Let plaintext = 0b0110 and key = 0b1101. (Note: in Python, you can define an integer by using e.g. x = 0b1111 (which will create a variable called x with the value 15). Also note that XOR is only defined between two integers.

(a): Compute the XOR of plaintext and key, call this the ciphertext.

(b): Compute the XOR of ciphertext and the key. What do you observe?

(c): Change one bit of the ciphertext and compute the XOR of it and the key. What happens to the plaintext? (Note: This property is called malleability.)

Exercise 2: Prove that XOR encryption is a valid encryption method. (Hint: You need to prove that given an arbitrary plaintext and key, encrypting the plaintext given the key and then decrypting the ciphertext using the same key will bring back the plaintext.)

Block Ciphers

Exercise 3: Starting from the AES-ECB live coding example, add functions so that the user can input an arbitrary text string. This text string should then be encrypted, and the ciphertext should be printed in the console. Be sure to check that your code can decrypt the ciphertext also. (Hint: Note that the code requires plaintext and ciphertext to be bytearrays.)

Exercise 4: Modify the code from the previous exercise so that the user can choose whether to provide a key or have the program generate a key. Try to encrypt with simple keys such as the all-zeros bitstring.

Exercise 5: Consider the following key and nonce, here in their Base64 representation:

Key: WXvqxbAEEM08snXbYgu8bg==

Nonce: ZmRqZW1ma3Jtc2thbXNuZA==

Decrypt the following ciphertext (here in Base64) using AES in CTR mode:

M8owlYB5ewJJ2YFcdLdXJu0DHlv4+9iqUpdpJeRyH3SRqg==

(Hint: to convert from Base64 to bytearray, have a look at the Base64 module in Python)

Exercise 6: In the previous exercise, how many blocks does the ciphertext have. Does the CTR mode use padding (why/why not?)

Exercise 7: Note that CTR uses both a nonce and counter to encrypt data. However, you were only given the key and nonce (and not the counter). What does that mean for the value of the counter?

Stream Ciphers

Exercise 8: In this exercise, we are working with the stream cipher ChaCha20 in Python. See https://cryptography.io/en/latest/hazmat/primitives/symmetric-encryption/#cryptography.hazmat.primitives.ciphers.algorithms.ChaCha20 for usage and an example.

(a): Encrypt the plaintext “Hello world” (without quotes) using the stream cipher ChaCha20. Be sure to generate a key and nonce properly. Print the ciphertext in the console, what do you see?

(b): Decrypt the ciphertext obtained in part (a), and check that the original plaintext and decrypted ciphertext are equal (why do we want to check this?)

(c): Change one byte of the ciphertext and decrypt it using the same key and nonce as in part (a). What do you observe?

(d): Change one byte of the key and decrypt the ciphertext obtained in part (b). What do you see?

If you have finished the above exercises, you can look at the ones from Set 1 in Cryptopals: https://cryptopals.com/sets/1. You can start with challenge 2 and afterwards, challenge 3.