Homework Assignment 2
Information Security - Theory Vs. Reality
0368-4474
Submit by: 24th of May at 13:59

1 Submission Instructions

The HW assignments will include writing code and answering written questions. The submission will include the requested code file and a PDF file with the written answers. The PDF can be a scan of a *clearly handwritten* page, but typing the answers is *strongly* encouraged.

The zip file should be of type .zip and not .rar.

Unless stated otherwise, all assignments must be written in Python 3.8.

We set up a docker environment with python3 all required packages to allow you to run and test your code. You can use other developing environments for writing your code, but you need to make sure it runs on the python version installed in the docker before submitting it.

Code that fails to run inside the docker environment will not be graded!

Instruction for running the python3 environment:

1. Login to nova or any cs server of your choice.
2. Run the following 2 commands to start the docker:
   `export UDOCKER_DIR="/specific/netapp5_2/eyalron1/SecCourseDocker"`
   `udocker run --bindhome SecDock`
3. If all works well, you should now be running inside the docker, with the docker's home directory mapped to your own home directory. Note that you can only save files inside your home directory.

4. You can run python with the command:
   `python3`

2 Coding Assignment 2

For your second coding assignment, you are requested to implement two attacks on AES.

2.1 Leakage from T-Table

The first will exploit the data leaked from cache attack on T-Table implementation of AES. In this attack, you are given a function that simulates the data that can be recovered from accesses to the cache during T-Table based AES encryption, as presented in class. You are then expected to exploit that data to recover the full encryption key. For this attack, please implement the missing parts (denoted by "?") in the file `aes_ttable_cache_attack.py`. The function `cache_attack` should return a list of candidate keys, where each key is of type bytes.

2.2 Leakage from Hamming Weight

The second attack will exploit data the is leaked from a power analysis attack. Here we assume that measuring the power output of the encryption process allows you to know the Hamming distance between the input and the output of each S-box during the encryption process, for example:

$$HW(S[p[0] \oplus k[0]] \oplus p[0] \oplus k[0])$$

. You are then expected to exploit that data to recover the full round key. For this attack, implement the missing parts (denoted by "?") in `aes_power_analysis.py`. The function `power_analysis_attack` should return a list of candidate keys, where each key is of type bytes.

2.3 Implementing the attacks

In both attacks you are expected to support keys of length 128, 192, and 256 bits. This means for example that if the `key_len` parameter in `cache_attack` is equal to 256, it should output 256-bit AES keys.

In order to help you test your work, both attacks have a function '`check_test_vectors`', which checks specific test vectors for several of the unimplemented functions. Take note that even if a function is implemented correctly, the test vectors may fail if the preceding functions are not correctly implemented.

If you prefer, you are permitted to implement the algorithms in a different manner than suggested by the skeleton code, as long as the functions `cache_attack` and `power_analysis_attack` still return the correct outputs.

3 Question 2

After you finish implementing both attacks:

1. Write a full description for each of the attacks you implemented.
2. For the T-Table based attack, describe what will be the effect on the attack if:
   (a) Instead of 64 bytes cache line we will have 4 byte cache line?
   (b) Instead of 64 bytes cache line we will have 8 byte cache line?
   (c) Instead of 64 bytes cache line we will have 128 byte cache line?
   (d) Instead of 64 bytes cache line we will have 1024 byte cache line?
3. For the Hamming Weight based attack, describe what will be the effect on the attack if:
   (a) Instead of leaking $HW(S[p[0] \oplus k[0]] \oplus p[0] \oplus k[0])$, the implementation will leak $HW(S[p[0] \oplus k[0]])$?
   (b) Instead of leaking $HW(S[p[0] \oplus k[0]] \oplus p[0] \oplus k[0])$, the implementation will leak $HW(p[0] \oplus k[0])$?