Homework Assignment 5
Information Security - Theory Vs. Reality
0368-4474
Submit by: 11th of Aug, 23:59

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.

If not written otherwise, all the files should be at the root of the zip (meaning that they are not inside an inner folder).

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 5

For your fifth coding assignment, you are requested to implement the following three algorithms:

2.1 Floyd's Algorithm

The first is Floyd's algorithm for collision detection. Here you are required to implement an algorithm that given a random function f (and a starting point) tries to find and return a collision in f.

For this section, please implement the missing parts (denoted by "?") in the file `floyd.py`.

2.2 Nivasch's Algorithm

The second algorithm is Nivasch's algorithm for cycle detection. Here you are required to implement an algorithm that given a random function f (and a starting point) finds a point on the cycle defined by f and the starting point. Implement the algorithm for a general amount of k stacks. The values x corresponding to stack i are those that hold $x \equiv i \; mod \; k$.

For this section, please implement the missing parts (denoted by "?") in the file `nivasch.py`. A test vector is integrated into the code to help you check your work.

2.3 Hellman's Time-Memory Tradeoff

The final algorithm is Hellman's Time-Memory Tradeoff. Here you are requested to implement two subroutines:

1. A subroutine that given a random function f generates preprocessed tables.
2. A subroutine that utilizes the above tables to find a value x such that $f(x) = y$ for a given value y.

You should handle three different cases:

- The case where the domain and range of f are equal.
- The case where the domain of f is smaller than its range.
- The case where the domain of f greater than its range.

This is done by implementing the class `ModifiedPRF` in the file.

Additionally, for each case choose parameters t and m according to the stopping rule of the algorithm:

$$mt^2 = N$$

For this section, please implement the missing parts (denoted by "?") in the file `hellman_tables.py`. A test for each case is integrated to help you check your work. Note that the algorithm doesn't necessarily succeed, so succeeding in a good amount of the tests is sufficient.

3 Question 5

Prove the correctness of Floyd's algorithm. Specifically, you are requested to prove that given the point when the two pointers collided, returning one of the pointers to the starting point and advancing both of them by one iteration each time, they will meet at the entry point to the cycle.