COMS 4181: Security I

Programming Assignment 1
Due October 15th, 11:59 pm

Late policy:
You have a total of two late days that you may choose to use in any of the assignments without any penalty. Any submission after the deadline will be considered as a late day and if it exceeds 24 hours after the deadline, then it will be considered as two late days. You may use the late days in separate assignments or together. Once you have used all of your late days, you will not receive any credit for late submissions.

Collaboration policy:
You are not allowed to discuss the solutions to homework problems/programming assignments with your fellow students.

Preliminaries:

A 64-bit Linux environment is required. If you do not have a Linux environment we recommend Ubuntu 18.04 as an option. If you choose to use VMs you can find ready to run images at OSBoxes (https://www.osboxes.org/ubuntu/) that will help you speed up the process. Alternatively, Debian should work as well.

The following packages should be installed (ie. Ubuntu/Debian):

- `apt-get install build-essential cmake python git xz-utils`

The following external dependencies will be downloaded & built automatically when you build the homework1 project and their use is allowed:

- Botan (https://botan.randombit.net/)

Extract the homework1 project from `homework1.tar.xz` using a command like the following:

```
$ tar xvf homework1.tar.xz
```

A folder named `student` should have been extracted.

To build your code run the following commands from inside the homework1 repository.

```
$ cd student
$ mkdir build && cd build
$ cmake ..
$ make
```

You will find all relevant binaries inside the `build/bin/part*` folders for each respective part.

***NOTE**:* Running `make clean` from the `build` folder will also clean the `botan` dependency forcing it to be rebuilt. If you'd like to clean a specific part cd into the appropriate directory clean from there. The following is an example:

```
$ cd build
$ cd src/part0
$ make clean
```

NOTE: Please ignore the `internal` namespace.

Please note you should not need to modify the `CMakeLists.txt` files. If you encounter any problems or have any questions, please post on Piazza. Please do not submit any code related to partial/full solutions on Piazza, they will be deleted.

Part 0: Detecting Block Cipher Properties (20 Points)

The class `EncryptionOracle` provides methods that you will be using (These can be found under `include/part*`). Under the hood this encrypts the plaintext with a given `KEY` and `IV`, randomly choosing between `ECB` & `CBC` modes.

You must detect two properties of block ciphers: block size and mode using the `encrypt()` method described above.

Please write your code in the corresponding functions in `src/part0/DetectionOracle.cpp`. You should then use the `main.cpp` in that folder to run your code. Sample code for invoking the oracle is provided in `src/part0/DetectionOracle.cpp.` Please, feel free to use it as a reference for other parts of this homework.

To be submitted:

1. `src/part0/DetectionOracle.cpp`

Part 1: ECB Decryption (20 Points)

The class `EncryptionOracle` provides an `encrypt` method that you will be using. Under the hood this method using a given `KEY` performs the following:
`AES-128-ECB(your-string + secret-string, KEY)` where + denotes concatenation.

You goal is to decrypt the `secret-string` that is appended before encryption.

Please write your code in the corresponding functions in `src/part1/ECBDecrypt.cpp`. You should then use the `main.cpp` in that folder to run your code.

To be submitted:

1. `src/part1/ECBDecrypt.cpp`

Part 2: CBC Decryption (10 Points)

The class `EncryptionOracle` provides an `encrypt` method that you will be using. Under the hood this method using a given `KEY` and `IV` performs the following:
`AES-128-CBC(your-string + secret-string, KEY, IV)` where + denotes concatenation.

You goal is to decrypt the `secret-string` that is appended before encryption.

Please write your code in the corresponding functions in `src/part2/CBCDecrypt.cpp`. You should then use the `main.cpp` in that folder to run your code.

To be submitted:

1. `src/part2/CBCDecrypt.cpp`

Part 3: ECB Decryption v2 (20 Points)

The class `EncryptionOracle` provides an `encrypt` method that you will be using. Under the hood this method using a given `KEY` and `IV` performs the following:
`AES-128-ECB(prefix + your-string + secret-string, KEY)` where + denotes concatenation.

You goal is to decrypt the `secret-string` that is appended before encryption.

Please write your code in the corresponding functions in `src/part3/ECBDecrypt.cpp`. You should then use the `main.cpp` in that folder to run your code.

To be submitted:

1. `src/part3/ECBDecrypt.cpp`

Part 4: CBC Padding Attack (20 Points)

The class `PaddingOracle` provides the following methods: `encrypt(), isValid()`. `encrypt()` will encrypt a secret string that you must find using the information leaked by `isValid()` (ie, whether the padding of the decoded message is valid or not).

Please write your code in the corresponding functions in `src/part4/PaddingOracleAttack.cpp`. You goal is to decrypt the `secret-string` that is appended before encryption.

To be submitted:

1. `src/part4/PaddingOracleAttack.cpp`

Part 5: Secure Crypto (10 Points)
Discuss how to remedy the attack in part 4.

To be submitted:

1. `src/part5/part5.txt`

Submission:

Please use the following command to generate an archive with the relevant code for submission:

```
$ make submission-archive
```

This will generate a file called `homework1_submission.tar.xz` which you can then submit on Courseworks.