Assignment 1

CS448 Introduction to Information Security, KAIST
(2021 Spring)

Due: 11:59 PM (KST), April 1, 2021

INSTRUCTIONS TO STUDENTS:

- Any collaboration or assistance of any kind is strictly prohibited; all work must be your own.
- Your submission *will surely* be compared with the submissions for your peers for plagiarism detection (e.g., MOSS). Any academic dishonesty will be directly reported to the university.
- You have total five grace days in this semester.

1 True-False Questions (30 points)

1. True or false: Message Authentication Code (MAC) prevents an adversary from modifying or injecting messages between two parties with a symmetric secret key; thus, it makes replay attacks ineffective.
   (A) True, (B) False
2. True or false: In the Needham-Schroeder protocol, each participating party (e.g., A or B) becomes an encryption oracle to the other; i.e., provide a ciphertext for a plaintext chosen by another party.
   (A) True, (B) False
3. True or false: All entity authentication protocols require on-line (i.e., always on, always accessible) trusted third parties.
   (A) True, (B) False
4. True or false: The Diffie-Hellman key exchange protocol needs a trusted third party to determine the group parameters used for shared-secret generation.
   (A) True, (B) False
5. True or false: Whenever $n = p \cdot q$, where p and q are distinct primes, finding Euler's totient function of n, denoted by $\phi(n)$, allows finding secret key d efficiently (i.e., in polynomial time in the length of n).
   (A) True, (B) False
6. True or false: A hash function that is second pre-image resistant is not necessarily pre-image resistant.
   (A) True, (B) False

1.1 Submission

Submission. Write a text file `p1_S_#.txt`, where # denotes your student id. In the text file, write your answer letters (i.e., A or B) for six questions and use `newline` to separate them. For example,

```
In p1_S_20210001.txt file:
    A
    B
    A
    B
    A
    B
```

Include this text file into the final tar file of your padding oracle attack; see the next section.

2 Padding Oracle Attack (70 points)

For this assignment, you will implement a padding attack discussed in class. This attack was first discussed in the paper "Security Flaws Induced by CBC Padding - Applications to SSL, IPSEC, WTLS..." by S. Vaudenay. You should read the paper (`http://dx.doi.org/10.1007/3-540-46035-7_35`) and implement the described attack.

You will have the opportunity to create your own padding attack discussed in class to extract a message from a ciphertext. You can find a set of valid ciphertexts on the KLMS assignment #1 handout section. The ciphertexts will be given as a 128-bit hexadecimal number (starting with `0x`...) computed as follows:

$$C = C_0 \| C_1 = \texttt{CBC}_K^{\texttt{Encrypt}}(\texttt{PAD}(M)), \tag{1}$$

where the word M is shorter than or equal to 8 ASCII characters (i.e., 64 bits). The block size of the cipher and the length of the secret key K are 64 bits. The first 64-bit block of C is the initialization vector (`IV`), which we denote C_0, and the last 64-bit block is the first cipher block, C_1. Note that we assume the randomized `CBC` encryption; that is, the `IV` (C_0) is randomly selected for every message.

You are given a *padding oracle*, which can be accessed by function calls; see Section 2.2 for the details. You can access the padding oracle as many times as you want. You will have two language options to develop your padding attack program: Java or Python. The padding oracle will accept both C_0 and C_1 (each 64 bits long), and return a 1 or a 0, indicating correct or incorrect padding respectively. The padding oracle works as follows:

$$\{0, 1\} = \texttt{Check\_PAD}(\texttt{CBC}_K^{\texttt{Decrypt}}(C)), \tag{2}$$

where you provide the C; i.e., you are a chosen-ciphertext attacker.

2.1 Where to obtain the ciphertext?

KLMS assignment #1.

2.2 How to access the oracle?

You will be given the following list of files:

- pad_oracle.jar: Padding oracle (Java bytecode).
- oracle_python_v1_2.py: Padding and decryption oracles (Python functions).
- python_interface_v1_2.jar: Java bytecode for interfacing Python scripts and oracle Java bytecodes.
- bcprov-jdk15-130.jar: Crypto library.

We recommend these versions for your assignment: Java 11, Python 3.6.
Here are the example codes that access the oracles.
Java:

```
// query padding oracle
pad_oracle p = new pad_oracle ();
boolean isPaddingCorrect = p.doOracle("0x1234567890abcdef", "0x1234567890abcdef");
```

Python:

```
Execute the Java-Python interface class by
java -cp pad_oracle.jar:bcprov-jdk15-130.jar:python_interface_v1_2.jar python_interface_v1_2
In your Python script,
from oracle_python_v1_2 import pad_oracle
# query padding oracle
ret_pad = pad_oracle('0x1234567890abcdef', '0x1234567890abcdef');
```

2.3 Submission and grading

Submission. Your code has name `p2_S_#.java` (or `.py`) where # denotes your student id. Tar all your codes and your True-False answer `p1_S_#.txt` (`S_#.tar`) and upload it to KLMS assignment #1. Your code should accept two command-line arguments C_0 and C_1 in the following format:

For java

- javac -cp pad_oracle.jar p2_S_#.java
- java -cp pad_oracle.jar:bcprov-jdk15-130.jar: p2_S_# C_0 C_1

For python

- python p2_S_#.py C_0 C_1

The ciphertext blocks (C_0 and C_1) have hex format starting with `0x`. The output must be the plaintext in ASCII format.

Grading. Your code should successfully obtain the plaintext from a ciphertext. You will get 40 pts if your code works for the ciphertexts that are available on handouts. If it works with all the other ciphertexts (not given to students), an additional 30 pts will be given.