CS 487 / 587 **Due: March 31 11:59pm**

(Bonus) Coding Homework 2

This project can give you up to 1% on top of your final grade.

Hello Folks!

In class we studied pseudorandom functions (PRF)s and CPA security. We discussed their definition of security and we saw some "bad" candidate constructions. For those "bad" constructions, we proved that indeed they do not satisfy the definition of security of PRFs.

Now, it is time to see play with PRFs in CPA security game using code! You will simulate a PRF game and you will try to win the game as an adversary (or a distinguisher or even a hacker). (I hope it excites you!)

Before getting started on the homework, you need to install Python and playcrypt library to be able to run the simulation. You can find the setup instructions, a coding example, and more details about the homework down below. Good luck!

Note: This homework is an optional bonus coding homework; it is not a requirement but we do encourage you to work on it - especially if you want to get a more "hands on" experience with certain crypto primitives. Students are welcome to work together on any of these problems, but *every student must write up their own solutions, independently!*

1 Required Installations

Every adversary needs to have some tools installed before cracking a system or a game. Since you don't have any coding homework before as a beginner adversary, please make sure that Python (preferably the latest version) and playcrypt library are installed on your machine. (You can use a Windows, Linux, or macOS)

- **Python:** You may want to check whether any of Python versions has already been installed in your machine by typing the following command in your terminal:

  ```
  python --version
  ```

  If you can see a version(preferably the latest version 3.11 but any version above 3 could work), this means that you have Python, yay! If you don't have it, you can use the official website to download and install Python.

- **playcrypt:** You will need to install the latest version of playcrypt. If you haven't yet installed it, you would install it as follows (if your default pip is not pip3, replace pip with pip3)

  ```
  pip install --user git+https://github.com/huseyingokay/playcrypt.git
  ```

  If you have an existing installation, you will need to uninstall it first

  ```
  pip uninstall playcrypt
  ```

2 Coding Example

Alright, our whole setup is ready to see a PRF - CPA security game! You can find the example game here, copy and paste the code to a Python file and run it on the terminal with the following command:

```
python cpa_example.py
```

The output of your code should look like the following output:

```
1.0
1.0
1.0
```

You may want to examine the example code by yourself first but the explanation of it is also provided down below!

```
11  def A(challenge, encQuery):
12      r"""
13      The adversary to the CPA Challenger.
14
15      :param challenge: a function pointer that allows the adverseary to make a query
16                        with a pair of plaintexts to be challenged on.  It returns a
17                        challenge ciphertext, and the adversary has to guess which of the
18                        plaintexts was encrypted in the challenge.
19      : param enc:      a function pointer that allows the adversary to make encryption
20                        queries.
21      :return: 0 if the first of the pair was encrypted, return 1 if the 2nd of the pair was encrypted.
22      """
23      c = challenge('0', '1')
24      y = encQuery('0')
25
26      if y == c :
27          return 0
28      else:
29          return 1
30
31  # DO NOT EDIT THIS FUNCTION #
32  def encrypt(k, m):
33      """
34      This uses a secure block cipher (PRF) b, and key k to encrypt message m
35      """
36      return b.encode(k, m)
37
38
39  # Use this block for any printing/testing, do not leave stray print
40  # statements outside this block.
41  if __name__ == "__main__":
42
43      keyLen = 128
44
45      b = BlockCipher(keyLen, 1)
46      g = GamePrivKCPA(1, encrypt, keyLen)
47
48      s = PrivKCPASim(g, A)
49      print(s.compute_success_ratio(0,1024))
50      print(s.compute_success_ratio(1,1024))
51      print(str(s.compute_advantage(1024)))
```

- We have three main parts in the example: A (adversary) method [lines 11-29], encrypt method[lines 32-36], and **the main section** [lines 41-51]
- A (adversary) is the method where you need to code your logic to break the challenge which sent by the challenger(basically, the output of encrypt method). **The only method you need to complete in your homework is "A" method**. Let's see how our example adversary breaks the challenge! (Don't forget that adversary knows the logic/algorithm of the challenger)
  - First, adversary chooses two messages "0" and "1" and sends to the challenger to get the challenge string c. (line 23)
  - Then, adversary requests to encrypt the string "0" from the challenger and gets the ciphertext y (line 24)
  - Adversary checks if y is equal to c. If so, adversary returns 0(which means the adversary decided that the first message is encrypted by challenger). Otherwise, adversary returns 1(which means the adversary decided that the second message is encrypted by challenger) (lines 26-29)

As a summary, adversary selects two strings and requests the ciphertext of their first string. Eventually, adversary checks whether if the ciphertext of the first string is equal to the challenge string.

- encrypt method is basically the function that the challenger uses to encrypt messages sent by the adversary. A pseudorandom function(PRF) b is used to encrypt the given message as following:

$$b.encode(k, m)$$

and where k is the key to choose the function and m is the message given by adversary. The code line above can be expressed as:

$$F(k, m)$$

where F is a pseudorandom function(PRF).

3 Homework

We have seen an example of a PRF - CPA security game and it is your turn now to be an adversary!

- Complete the playcrypt assignment by filling in the code for the adversary (stater code is provided on blackboard under the file cpa1_playcrypt.py). Your output should look like the following output:

```
0.521484375 (0.52...) (0.5 + q(n)/2^n)
0.998046875 (0.99...)
0.74951171875 (0.74..) (average of above)
```

In the output, $q(n)$ is the number of queries where the same random value is used in the encryption.

- The formal definition of encrypt method is as follows:

$\mathsf{Gen}(1^n) : k \leftarrow \{0,1\}^n$
$\mathsf{Enc}(k, m)$: Sample $r \leftarrow \{0,1\}^n$.
If $F(k, m)$ ends in '000', output $(r, F(k, m))$
Otherwise, output $(r, F(k, r) \oplus m)$.

In the encryption method (names as encrypt), challenger picks a random value r. Then, challenger gives the message to the pseudorandom function F as $F(k, m)$:

  - If the output of $F(k, m)$ ends with "000", then the challenger outputs the $(r, F(k, m))$ pair (You can access r and $F(k, m)$ separately in the output of encrypt method)
  - Otherwise, the challenger outputs $(r, F(k, r) \oplus m)$ pair where it basically XORs the output of $F(k, r)$ and the message m in the second element.

As the adversary, you can request the ciphertext of any message by using *encQuery* method. Eventually, you need to give two messages that you choose by using *challenge* method, get the challenge string, and determine which message you chose is encrypted by the challenger.

Good Luck!

4 Submission Format

You need to submit a Python file named as "[your name]_[your surname]_cpa.py" (i.e. john_smith_cpa.py) on Blackboard. Please just complete "A" method and delete your code lines which are for debug purposes (basically your print statements)