Homework 4: Cryptography I

Due March 10, 11:00 PM

---

1 Homework Overview

Each student is given a different collection of 12 quotes. We have created a unique secret key for each student and used it to encrypt the quotes. We have encrypted each quote separately using the one-time pad method. The length of the key is the same as the length of a single quote and thus, no padding was required. We have used the same key for all 12 quotes. This of course makes our ciphertexts susceptible to attacks. Your objective is 1) to write some helper code and then 2) to decipher the 1st quote.

2 Getting Started

Starter files are here.
From these files you will need:

- task1.py
- The folder named as your university directory (i.e. the username you use to log into university services) from the appropriate section folder. This folder contains your 12 encrypted quotes, one per file.

3 Plaintext Format

- Each quote consists of a single sentence.
- The length of each sentence is exactly 60 characters.
- There are no punctuation marks. All characters are either letters of the alphabet or spaces.
- The 60^{th} character might be a space.
- The first character is always uppercase.
- The rest of the characters are lowercase except for the personal pronoun I.
- There are no names or obscure words included.

Since each quote is abruptly cut off after the 60^{th} character, don't expect to get much wisdom out of it.

4 Ciphertext Format

- When a byte is encypted with a random key, it can take any value between $\{0, 2^8-1\}$. Thus, there is a high chance that the resulting value will not be a valid ASCII code and won't be readable with a standard text editor. For your convenience, we have converted all ciphertexts in hexadecimal format using the ASCII characters 0-9 and a-f. Note that since each hex digit represents 4 bits, each encrypted quote is now 120 characters long instead of 60.

- The downside is that you have to convert back to the original raw binary data format if you want to apply binary operations (like XOR). You also need to be in raw binary format when you want to see the plaintext in English.

5 Task 1

We have provided a file *task1.py* with a stub function `def task1(c1bin, c2bin, guessbin, offset)`. In this task you will finish this function and make task1.py into a tool that helps with breaking the encryption of the target ciphertext. Building it is meant to guide you on your way in task 2.

The function takes as input two ciphertexts and a guess for the contents of one of the two ciphertexts. It computes the resulting decryption of the other ciphertext assuming the guess was right. What the function returns should always have the same length as the lengths of the ciphertexts. The guess can be partial (i.e. not as long as a full ciphertext) and the offset clarifies the exact position of the guess against the ciphertext. The given code is unfinished and you must modify the function to achieve the described behaviour. What does that mean? You need to figure that out on your own.

As a sanity check, when `task1(c1bin,c2bin,guessbin,offset)` is finished properly, if your guess is correct and the full length of the quote (i.e. you have guessed correctly all characters of the quote), then you will have decrypted the other ciphertext. That is, the function will return the quote in the other ciphertext. In the likely case that your guess is only partially right, or you only guessed a short fragment, then part of the output will be correct but the rest of the output will be garbage. If your guess is wrong, you will output a meaningless message. In all cases the output will be the same length as the ciphertext. For task1 it does not matter if your guesses are correct, the point is for the function to go through the steps you figured out.

Formal Tips

- task1.py is a python 3 file. It can be run as:
  `python3 task1.py`
- You should return a bytearray or a bytes object. Output is already initialized to a bytearray of the proper size.
- For this task, do not install or make use of extra libraries outside of the python standard library. The system the autograder runs on will not have these libraries and your code will get a zero. The python 3 standard library https://docs.python.org/3/library/ is extensive and will work. You shouldn't need anything else for this task.

6 Task 2

You must decrypt the first ciphertext and thus retrieve the first quote. The finished code from task 1 can be very helpful but there are many ways to approach this problem. Some of these techniques can be used in conjunction with the requested function and some are standalone. Feel free to be creative and follow the path you desire. Full points will be awarded even if task 1 is not utilized, although you still have to accomplish both tasks.

7 What To Submit

Upload a tgz file on ELMS containing a folder with your directory id as the name. Inside the folder should be:

- The starter code file with a completed task1 function.
- Any other program that you wrote for task 2.
- A file named `my_guess.txt` that contains the 1st quote in plaintext.
- A brief pdf report where you describe the methodology and steps you followed in order to decipher.

8 Remarks

- Remember that $c_1 := m_1 \oplus k \land c_2 := m_2 \oplus k \rightarrow c_1 \oplus c_2 = m_1 \oplus m_2$
- If you take a careful look at the binary representation of ASCII characters, you should be able to observe a very unique relationship between space and the letters of the alphabet. This is vital for successful deciphering.
- It is not easy to create a fully automated solution with software. At some points pen and paper might be more effective.
- You may use your knowledge of English and common sense to guess characters.
- You may not use any outside programs. Any automated tooling you use to solve this must be your own. You may make use of reasonable libraries, but you can't just import someone's solver and call it a library.
- We will grade your guess character-by-character. For example, if the expected sentence was just the word *earthquake* and you guessed *earthquale*, that's 9/10 characters correct. But if you guessed *arthquacke*, that's 2/10 characters correct, since only the last 2 characters are properly aligned.