# Problem 4: Cryptonite [HackerRank]

By Nalin Ranjan

## 1 Background: The One-Time Pad

President Eisgruber is trying to send messages to Dean Dolan that he doesn't want anyone else to be able to read. He does some Googling and finds out about the *One-Time Pad (OTP)*, which according to Wikipedia, is an "encryption technique that cannot be cracked." To implement this, Eisgruber must first share a secret, random-looking key with Dean Dolan that is the same length as the messages he wishes to send. For example, if he wishes to send the message `APPLE`, encoded in ASCII, he must share a 40-bit key, because each character in ASCII takes 8 bits to encode. Then, he takes his message $m$ and his secret $s$ and does a bitwise exclusive-or (XOR) between them to produce a ciphertext, which he can then broadcast publically. For example, if his secret key is `i<3me` and the message he wishes to send is `APPLE` (both encoded in ASCII), then the ciphertext that he sends to Dean Dolan would be `(lc!`$_s$ (the last character is a whitespace):

```
        i         <         3         m         e

    01101001  00111100  00110011  01101101  01100101
 ⊕  01000001  01010000  01010000  01001100  01000101
        A         P         P         L         E
 ----------------------------------------------------
    00101000  01101100  01100011  00100001  00100000
        (         l         c         !        " "
                                           (whitespace)
```

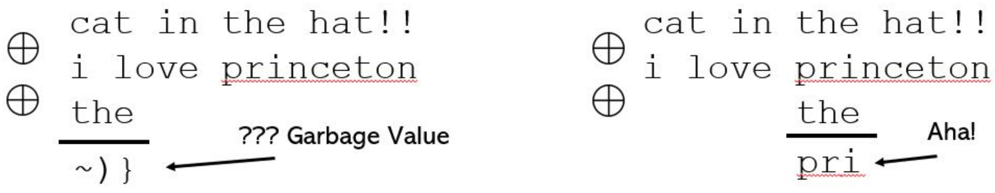
(The ⊕ symbol is the exclusive-or operation.) To recover the plaintext message, Dean Dolan just needs to XOR the ciphertext she receives with the secret key.

Unfortunately, it's been 42 years since he took Writing Seminar, and he needs to brush up on his close reading skills... because he didn't notice that Wikipedia also says that the "key must never be reused in whole or in part"! If he tries to do this scheme with two messages $m_1$ and $m_2$, to create two ciphertexts $c_1 = m_1 \oplus s$ and $c_2 = m_2 \oplus s$, then what happens when an evil adversary XORs these ciphertexts? Well, the XOR operation is both commutative and associative, so

$$(m_1 \oplus s) \oplus (m_2 \oplus s) = (m_1 \oplus m_2) \oplus (s \oplus s) = m_1 \oplus m_2$$

because any value XORed with itself is just all zeroes, and any value XORed with all zeroes is just itself.

Let us refer to $m_1 \oplus m_2$ as $x$ from now on. If we further know that $m_1$ and $m_2$ are plaintext English and encoded in ASCII, for example, then we can start guessing a word or phrase that is likely to be in the message (called the "crib") and "drag" it across $x$. If we are lucky and it is indeed a part of one of the ciphertexts, then when we reach the exact place in the $x$ corresponding to the place where the phrase is located in one of the plaintexts, an XOR of $x$ and the crib will yield something intelligible! Otherwise, it is overwhelmingly likely that an XOR of $x$ and the crib yields some unintelligible, garbage value. For example, if $m_1 =$ `cat in the hat!!`, $m_2 =$ `i love princeton`, and our crib is `the`, then when we XOR `the` with the eighth, ninth, and tenth characters of $x$, we notice that the result is the `pri`, while if we XOR `the` with any other three consecutive characters of $x$, the result is garbage:



This then gives us information about the second message, as we now know that `pri` is part of the second message. We then could guess that the `pri` is part of `print`, `princess`, `capricious`, or even the right answer, `princeton`. This process can be continued, uncovering more and more characters of the plaintext messages until it becomes obvious what the messages say. Feel free to read more about the OTP on Wikipedia, or come to our Zoom room to ask!

## 2 Problems

In these problems, you will be cracking a couple variants of the OTP. Make sure to download the accompanying files from our website. We've provided the ciphertexts in two formats: 1) as readable binary files, where you can see the ones and zeroes and 2) plain binary files, which are easier for programs to parse. The readable binary files have the suffix `readable`, while the plain binary files have the suffix `plain`. We've provided, under the `binary_reader` directory, files that help you read in the ciphertexts as binary data. We've accommodated three languages: Java, Python, and C++. Feel free to use the code we've given you so you don't have to pour over reading binary data for too long.

(i) (6 points) Eisgruber finds the OTP too “bland” for his taste and decides to tweak it. His encryption of the $i$th message will be $c_i = (s + i) \oplus m_i$, where $s$ is his secret key and $m_i$ is the $i$th plaintext message. Under the `p1` subdirectory are three consecutive ciphertexts — $c_1, c_2$, and $c_3$ — that he sent. What do they say?

In this part, you may assume that each character in the plaintext messages is an ASCII printable character, and furthermore, each letter will be lowercase. This means each character will have decimal code in the range 32-64 or in the range 91 to 126 (both ranges are inclusive).

(ii) (9 points) After you cracked Eisgruber's original encryption and posted his sensitive messages on Tiger Confessions, he decides to beef up his encryption scheme (or so he thinks). In the depth of night, Eisgruber sneaks past PSAFE and delivers two new secret keys, $s_1$ and $s_2$, to Dean Dolan's office. The next day, he proudly announces to the world that the encryption of the $i$th message will be $c_i = (s_1 + i \cdot s_2) \oplus m_i$. Again, $m_i$ is the $i$th plaintext message.

Instead of us giving you a fixed number of ciphertexts and asking you to decode them, this time we've only given you one ciphertext to start off with (this can be found in the `p2` subdirectory), and it is on *you* to request up to nine additional ciphertexts from us to try and crack. To get full credit for this part, you must only fully crack only *one* of any of the ciphertexts you receive. If you find that you want access to more ciphertexts, you can always tell us request more on our problems page. For this part, you can assume that every character is either a lowercase letter, or has decimal ASCII code in the set $\{32, 33, 39, 44, 46, 63\}$. The codes in this set correspond to the whitespace, exclamation mark, apostrophe, comma, period, and the question mark.

But wait, you might ask, why would we voluntarily choose to do *more* work cracking *more* ciphertexts? Good question! It turns out that the more you have, the easier it becomes to crack anyone of them. For this reason, we will be giving a special prize to the team that can complete the question with access to the *fewest* ciphertexts.

For all three parts, you will be submitting your decrypted plaintexts on Hackerrank (more specifically, you will just write programs to print out the decrypted plaintexts).

**Note**: The addition and multiplication operations described in the problems are all done modulo $2^L$, where $L$ is the message length. This is equivalent to calculating the result of the arithmetic formula, then only taking the $L$ least significant bits and XORing it with the plaintext.