TP1 Report

A85272	Jorge Mota
A83840	Maria Silva

The had the goal to identify the 2 ciphertexts of the 20 ones provided, that used the same key for OTP (One Time Pad) encryption.

Unfortunately can't say for sure that this work was completed successfully, since the only guess we have we can't confirm. If the letter frequency in the plaintexts texts match the values we assumed (english alphabet letter frequency).

In this *One Time Pad* instead of XOR of the bytes the operation was oriented to the addition of alphabetic characters with rotation on overflow (mod 26).

```
'A' + 'B' = 'B' 'B' + 'C' = 'D' 'Z' + 'B' = 'A'
```

Our Attempt

With 20 ciphertexts we wanted to find any direct relation between 2 texts, for this we first tried to subtract ciphertext1 (c1) and ciphertext2 (c2) (for every possible pair in the 20 texts).

``` c1: ciphertext 1 c2: ciphertext 2 t1: plaintext 1 t2: plaintext 2 k: reused key k1: random key 1 k2: random key 2

Case A Case B With same key With random keys c1 = t1 + k c1 = t1 + k1 c2 = t2 + k c2 = t2 + k2

c2 - c1 = (t2 + k) - (t1 + k) c2 - c1 = (t2 + k2) - (t1 + k1) c2 - c1 = (t2 - t1) c2 - c1 = (t2 - t1) + (k2 - k1) ``` As we can see with same keys the subtraction of the ciphertexts cancel the adition of the keys and the result is only the diference between text 1 and 2.

On the other hand the subtraction of ciphers with random keys results in the diference between text 1 and 2, and the aditional subtraction of random keys. This is a randomness feeder for the end result.

With this we can say that in the case B the `c2 - c1` will have a extra factor of randomness.

In the English alphabet we have diferent letter frequencies. Assuming that `t1` and `t2` have a similar letter frequency to the table we listed below, we can assume that the probability that the subtraction of 2 equal letters is higher for english texts than for random strings.

**Letter Frequency**

| E | T | A | O | I | N | S | R | H | D | L | U | C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 12.02 | 9.10 | 8.12 | 7.68 | 7.31 | 6.95 | 6.28 | 6.02 | 5.92 | 4.32 | 3.98 | 2.88 | 2.71 |

| M | F | Y | W | G | P | B | V | K | X | Q | J | Z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2.61 | 2.30 | 2.11 | 2.09 | 2.03 | 1.82 | 1.49 | 1.11 | 0.69 | 0.17 | 0.11 | 0.10 | 0.07 |

Since the Case B has a randomness feeder with the addition of the subtraction of diferent keys, the `c2 - c1` of Case A will have a higher chance of having the letter `A` than Case B.

Finally we tested this with python code to count letter `A` occurrences to verify this theory and sorted the results.
```

Results

```
 $ python main.py | sort . . . 231 (15,20): 232 (4,13): 240 (7,20): 244 (12,20): 248 (3,14): 371 (6,14): Highest: (6, 14, 371)
```

Analyzing the results we can verify that the subtraction of ciphertexts **6** and **14** had more `A` occurrences than any other ciphertexts combinations and thus making us believe that these were the two that were encrypted with the same key.

However, we took the assumption that the letter frequency in both plaintexts matched with the letter frequency in the english alphabet. In case this isn't true then the result might not be correct and the reason the number 371 is so high is just coincidence.