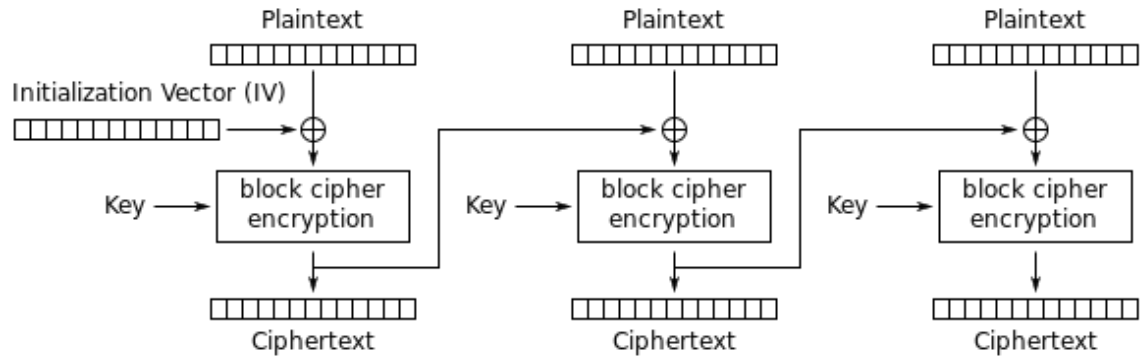
Problem #1 (Breaking of a Classical Cipher)

1. Explanation of encrypt_classical.py

Major Highlights

- Encryption used is CBC(Cipher Block Chaining)
- In CBC, each block of plaintext is XORed with the previous ciphertext block before being encrypted,
  and then it is xored with the same key of size == Block size
- Zero Padding is used at the last block, if the plain text is not completely fit into Blocks, extra padding is used at the end.
- Initial Vector(IV) is used as passphrase.
- Block size used is 64 bits == 8 bytes == 8 CHARACTERS
- Key size is same as Block Size == 64 bits

For more further study, read here



Cipher Block Chaining (CBC) mode encryption

Line by line explanation

- Line 8-9: import libraries
- Line 21-25: Reduce the passphrase to a bit array of size BLOCKSIZE
- Line 28-33: Get key as user input
- Line 36-39: Reduce the key to a bit array of size BLOCKSIZE
- Line 45-54: Xoring the Blocks as follows
  C(i) = C(i-1) ^ P(i) ^ K, where i ∈ [0, NUMB_BLOCKS]
  For i = 0, C(i-1) is Initial Vector
- Line 57: Convert the Bitvector to a ascii charater string
- Line 60-62: Save the obtained cipher text to file.

2. decrypt_classical.py

This will take 2 command line argument

- The ciphertext file
- Output file, where you want to store plaintext.

It will then ask for key, by which you encrypted.

```
$ python decrypt_classical.py encrypt_txt.txt  out_plain.txt
```

```
Enter key: bits@f463
```

3. crack_classical.py

In the crack_classical, I have first implemented Oracle Padding attack, which checks for the zero-padding in the last block, if there is padding available in the last block, then we can easily get the key block out of it.

So, amount of zero-padding, will reveal that many size key.

This key is then supplied to Many time pad attack, as same key is been used in all the blocks. So we can use the Space attack which is been briefly discussed in problem #2 attack.

For applying the above attack, we have to first modify the ciphertext as follows.

```
C(i) = C(i-1) ^ P(i) ^ K, where i ∈ [0, NUMB_BLOCKS]
==> C(i-1) ^ C(i) = P(i) ^ K
==> C'(i) = P(i) ^ K
==> P(i) = C'(i) ^ K
```

After then we can think of Number of blocks ciphertexts, which are encrypted using same key.

Then the following space attack.

```
Space XOR [.] = [.]
Space XOR Space = 0

X XOR Space = x; where X = [A-Z], x = [a-z] and vice versa
```

How to run?

```
$ python crack_classical.py
```

```
Enter filename of cipher text to crack: ciphertext.txt
Crack started!!

Plain text is written into recoveredtext.txt file.
```

4. Assumptions and Limitations

- The characters for which key character is not found, I have putted `*` at that position in the `recoveredtext.txt` file. These `*` means these characters are not revealed in this attack.

However many of the words can be completed by obvious guessing.

- The attack is dependent on size of cipher text available, more is the cipher text size, more the characters of plain text is revealed.
- It is been seen that, if the cipher text is more than ~30 Cipher blocks i.e., > 180 characters, then we can reveal whole plaintext.
- In the oracle padding attack, I am assuming that 7 padding (out of 8 available), is done at last cipher block. However this key is then further filtered into 2nd attack which is MTP attack.
- If the cipher text size given is less than expected, then partially plain text is revealed.