Breaking of a Classical Cipher

BITS F463: Cryptography
AY 2022-23, II Semester

Dhruv Rawat (2019B3A70537P)

April 30, 2023

The encryption script in `encrypt_classical_p1.py` is based on **differential XORing** of bit blocks. The plaintext file is scanned in blocks of 64 bits and the output ptoduced for each block is made a function of output of the previous block.

The encryption script requires a `key` and a `Passphrase`
Here, `key` = `bits@f463` and `Passphrase` = `Cryptography is the art of secret writing`

The `Passphrase` is used as an **Initialisation vector (IV)**.

The plaintext fileis scanned in bit blocks, with each block being of size `BLOCKSIZE`. Here the `BLOCKSIZE` has been taken as 64. Since the size of the plaintext might not be an integral multiple of `BLOCKSIZE`, appropriate number of 0 bytes are appended to the last block to make its size same as `BLOCKSIZE`.

Each bit block read from the plaintext file is first **XOR**ed ($\oplus$) with the `key` and then with the output produced for the previous bit block.

Let b_i represent blocks of plaintext and B_i represent the corresponding blocks of ciphertext

$$
\begin{aligned}
B_1 &= b_1 \oplus \texttt{key} \oplus \texttt{IV} \\
B_2 &= b_2 \oplus \texttt{key} \oplus B_1 = b_2 \oplus b_1 \oplus \texttt{key} \oplus \texttt{IV} \\
B_3 &= b_3 \oplus \texttt{key} \oplus B_2 = b_3 \oplus b_2 \oplus b_1 \oplus \texttt{key} \oplus \texttt{IV} \\
B_4 &= b_4 \oplus \texttt{key} \oplus B_3 = b_4 \oplus b_3 \oplus b_2 \oplus b_1 \oplus \texttt{key} \oplus \texttt{IV} \\
B_5 &= b_5 \oplus \texttt{key} \oplus B_4 = b_5 \oplus b_4 \oplus b_3 \oplus b_2 \oplus b_1 \oplus \texttt{key} \oplus \texttt{IV} \\
&\vdots \\
B_n &= b_n \oplus \texttt{key} \oplus B_{n-1} = b_n \oplus b_{n-1} \oplus b_{n-2} \oplus ... \oplus b_3 \oplus b_2 \oplus b_1 \oplus \texttt{key} \oplus \texttt{IV}
\end{aligned}
$$

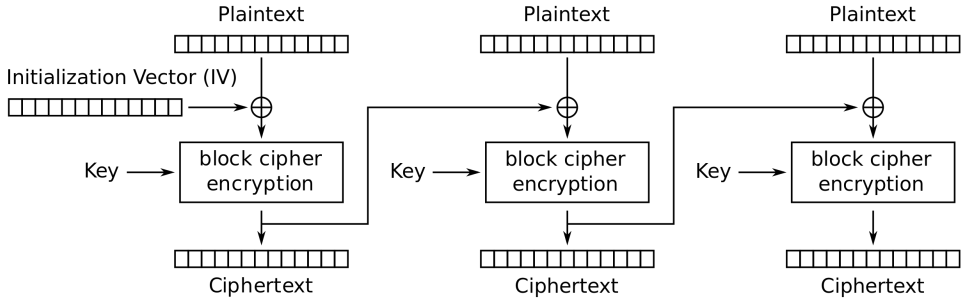
$$
\boxed{C = B_1 \| B_2 \| B_3 \| B_4 \| B_5 \| ... \| B_{n-2} \| B_{n-1} \| B_n} \tag{1}
$$

CBC (Cipher Block Chaining) is a symmetric encryption mode that operates on fixed-size blocks of plaintext. CBC mode is one of the most commonly used encryption modes, and it is widely used in various applications like SSL/TLS, IPSec, and others.

In CBC mode, each plaintext block is first XORed with the previous ciphertext block before being encrypted with the secret key. This chaining of the ciphertext blocks ensures that even small changes in the plaintext result in major changes in the ciphertext.

Here are the steps for CBC encryption:

1. Divide the plaintext into fixed-size blocks.
2. Generate a random initialization vector (IV).
3. XOR the first plaintext block with the IV.
4. Encrypt the result of step 3 with the secret key to obtain the first ciphertext block.
5. XOR the second plaintext block with the first ciphertext block.
6. Encrypt the result of step 5 with the secret key to obtain the second ciphertext block.
7. Repeat steps 5-6 for all the remaining plaintext blocks.

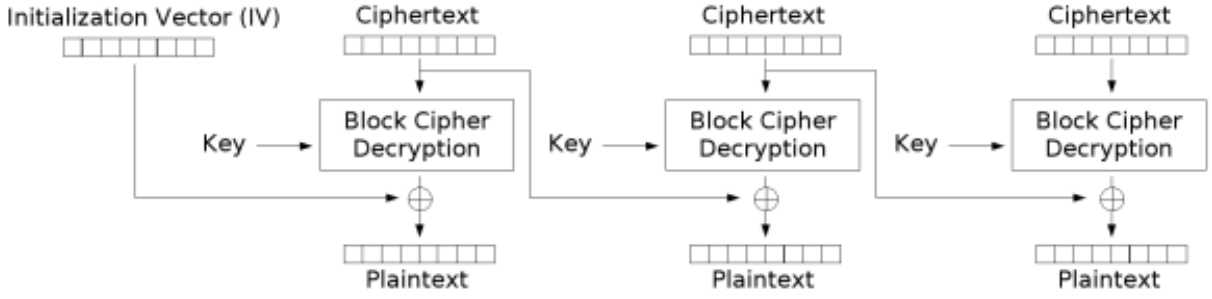


Cipher Block Chaining (CBC) mode encryption

To decrypt the ciphertext in CBC mode, the same key used for encryption and the initialization vector must be used.

Here are the steps for CBC decryption:

1. Divide the ciphertext into fixed-size blocks.
2. Decrypt the first ciphertext block using the secret key.
3. XOR the result of step 2 with the IV to obtain the first plaintext block.
4. Decrypt the second ciphertext block using the secret key.
5. XOR the result of step 4 with the first ciphertext block to obtain the second plaintext block.
6. Repeat steps 4-5 for all the remaining ciphertext blocks.



Cipher Block Chaining (CBC) mode decryption