Assessment 03 - Block ciphers

Instructions
The assessment must be developed individually and submitted via Blackboard by **24/10/2025**. The submission should be a *zip* file containing all the developed code and a short report in *PDF* format describing the developed code and discussing the achieved results.

Activities

Most encryption modes of block ciphers require an Initialization Vector (IV), which is not supposed to be a secret. However, it must be unique, which means that no IV may be reused under the same key. To illustrate this, let's consider the *known-plaintext attack* model used for deciding whether an encryption scheme is secure or not. In this model, an attacker has access to both the *plaintexts* and *ciphertexts*. If this can lead to the revealing of further secret information, the scheme is considered insecure.

Q1: Based on the following code that implements the AES cipher in the OFB operation mode, discover the *unknown plaintext* based on the known pair of *plaintext* and underlying *ciphertext*. Assume that both *ciphertexts* were generated using the same *key* and *IV*.

```
Known pair:
Plaintext: "This is a message you can see the content"
Ciphertext:
b'G\xa4\xff\xc9\x05\x15\xf6\x91\xd6\x00\xfe_k\x1e$\x93o\xdd\x1bH\xa8\xb1\x89\xc3\xfa\xac^n\xb5\xc7\x91\x9
f5\x9bq7J\xafB\xecp'

Unknown message:
Ciphertext: b'J\xa3\xe3\x9aV\x14\xea\xc4\xdbD\xb3TwM.\x9ae\x8aBS\xb5\xf8\x99\x82\xfb\xe2H'
```

Q2: Now, consider you do not have access to a selected pair of known *plaintext* and *ciphertext*. Is there a way to break the AES-OFB when the *key* and *IV* are reused? If so, demonstrate how to do it.

```python
# AES-OFB code
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
from cryptography.hazmat.backends import default_backend

def aes_ofb_encrypt(key: bytes, iv: bytes, plaintext: bytes) -> bytes:
 cipher = Cipher(algorithms.AES(key), modes.OFB(iv), backend=default_backend())
 encryptor = cipher.encryptor()
 ciphertext = encryptor.update(plaintext) + encryptor.finalize()
 return ciphertext


def aes_ofb_decrypt(key: bytes, iv: bytes, ciphertext: bytes) -> bytes:
 cipher = Cipher(algorithms.AES(key), modes.OFB(iv), backend=default_backend())
 decryptor = cipher.decryptor()
 plaintext = decryptor.update(ciphertext) + decryptor.finalize()
 return plaintext
```