

19. Write a python program for encryption in the cipher block chaining (CBC) mode using an algorithm stronger than DES. 3DES is a good candidate. Both of which follow from the definition of CBC. Which of the two would you choose: a. For security? b. For performance?

**Code:**

```
#!/usr/bin/env python3
```

```
"""
```

3DES-CBC encryption/decryption using the system 'openssl' command.

No Python crypto modules required.

Requirements:

- 'openssl' must be installed and on PATH (common on Linux/macOS; on Windows install OpenSSL).

- Python 3.6+

This script:

- generates a 24-byte (192-bit) 3DES key (3-key 3DES)
- generates a random 8-byte IV (DES block size)
- encrypts plaintext (binary) with 3DES-CBC using openssl
- decrypts back and verifies

Notes:

- We call `openssl enc -des-ede3-cbc -nosalt -K <hexkey> -iv <hexiv>`
- OpenSSL by default applies PKCS#7 padding for block modes.

```
"""
```

```
import secrets
import subprocess
import binascii
import sys
from typing import Tuple

def gen_3des_key_bytes() -> bytes:
    """Generate a 24-byte 3DES key (3 independent 8-byte keys)."""
    return secrets.token_bytes(24)
```

```
def gen_iv() -> bytes:
    """DES/3DES block size is 8 bytes."""
    return secrets.token_bytes(8)

def bytes_to_hex(b: bytes) -> str:
    """Return lowercase hex (no 0x) for openssl -K/-iv."""
    return binascii.hexlify(b).decode('ascii')

def openssl_encrypt_3des_cbc(plaintext: bytes, key: bytes, iv: bytes) -> bytes:
    """ Encrypt plaintext using system openssl (3DES-CBC).
    Returns ciphertext bytes (binary).

    """
    key_hex = bytes_to_hex(key)
    iv_hex = bytes_to_hex(iv)
    cmd = [
        "openssl", "enc", "-des-ed3-cbc",
        "-nosalt", "-K", key_hex, "-iv", iv_hex
    ]
    proc = subprocess.run(cmd, input=plaintext, stdout=subprocess.PIPE,
    stderr=subprocess.PIPE)
    if proc.returncode != 0:
        raise RuntimeError(f"OpenSSL encrypt failed: {proc.stderr.decode('utf-8',
    errors='replace')}")
    return proc.stdout

def openssl_decrypt_3des_cbc(ciphertext: bytes, key: bytes, iv: bytes) -> bytes:
    """ Decrypt ciphertext using system openssl (3DES-CBC).
    Returns plaintext bytes (binary).

    """
    key_hex = bytes_to_hex(key)
    iv_hex = bytes_to_hex(iv)
```

```
cmd = [
    "openssl", "enc", "-des-ed3-cbc", "-d",
    "-nosalt", "-K", key_hex, "-iv", iv_hex
]

proc = subprocess.run(cmd, input=ciphertext, stdout=subprocess.PIPE,
stderr=subprocess.PIPE)

if proc.returncode != 0:
    raise RuntimeError(f"OpenSSL decrypt failed: {proc.stderr.decode('utf-8',
errors='replace')}")


return proc.stdout


def demo():

    message = b"Attack at dawn! This message is longer than one block."

    print("Plaintext:", message)

    key = gen_3des_key_bytes()

    iv = gen_iv()

    print("3DES key (hex):", bytes_to_hex(key))

    print("IV (hex):", bytes_to_hex(iv))

    ct = openssl_encrypt_3des_cbc(message, key, iv)

    print("Ciphertext (hex):", binascii.hexlify(ct).decode('ascii'))

    pt = openssl_decrypt_3des_cbc(ct, key, iv)

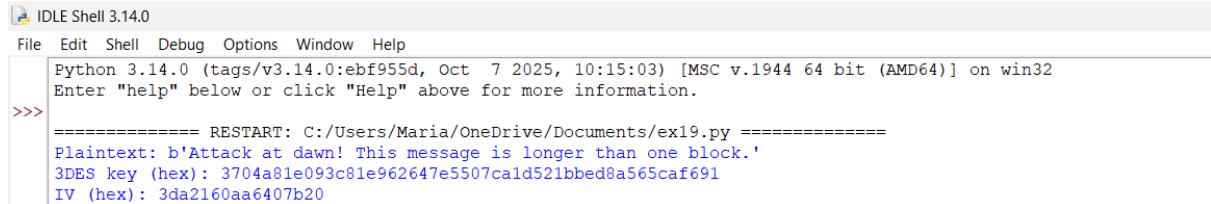
    print("Recovered plaintext:", pt)

    if pt == message:
        print("SUCCESS: Decrypted plaintext matches original.")

    else:
        print("FAIL: Decrypted plaintext differs!")

if __name__ == "__main__":
    try:
        demo()
    except FileNotFoundError:
```

```
print("ERROR: 'openssl' binary not found. Install OpenSSL and ensure it's on PATH.",  
file=sys.stderr)  
  
sys.exit(2)  
  
except Exception as e:  
  
    print("ERROR:", e, file=sys.stderr)  
  
    sys.exit(1)
```



The screenshot shows the Python 3.14.0 IDLE Shell interface. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the following text:

```
Python 3.14.0 (tags/v3.14.0:ebf955d, Oct  7 2025, 10:15:03) [MSC v.1944 64 bit (AMD64)] on win32  
Enter "help" below or click "Help" above for more information.  
>>> ===== RESTART: C:/Users/Maria/OneDrive/Documents/ex19.py ======  
Plaintext: b'Attack at dawn! This message is longer than one block.'  
3DES key (hex): 3704a81e093c81e962647e5507cald521bbcd8a565caf691  
IV (hex): 3da2160aa6407b20
```