Task 3

Advanced Incryption Tool

Code:-

```
    from cryptography.hazmat.primitives.ciphers import Cipher, algorithms,
modes
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import padding
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC
from cryptography.hazmat.primitives import hashes
import os
import base64

def derive_key(password: bytes, salt: bytes) -> bytes:
    """Derive a 32-byte AES key from the password and salt."""
    kdf = PBKDF2HMAC(
        algorithm=hashes.SHA256(),
        length=32,
        salt=salt,
        iterations=100000,
        backend=default_backend()
    )
    return kdf.derive(password)

def encrypt_file(password: str, filepath: str):
    """Encrypt the file using AES-256."""
    password = password.encode()
    salt = os.urandom(16)
    iv = os.urandom(16)
    key = derive_key(password, salt)

    with open(filepath, 'rb') as f:
        data = f.read()

    padder = padding.PKCS7(128).padder()
    padded_data = padder.update(data) + padder.finalize()

    cipher = Cipher(algorithms.AES(key), modes.CBC(iv),
backend=default_backend())
    encryptor = cipher.encryptor()
    encrypted_data = encryptor.update(padded_data) + encryptor.finalize()

    encrypted_file = filepath + ".enc"
    with open(encrypted_file, 'wb') as f:
        f.write(salt + iv + encrypted_data)

    print(f"File encrypted and saved as: {encrypted_file}")

def decrypt_file(password: str, filepath: str):
    """Decrypt the AES-256 encrypted file."""
    password = password.encode()
    with open(filepath, 'rb') as f:
        salt = f.read(16)
        iv = f.read(16)
        encrypted_data = f.read()

    key = derive_key(password, salt)

    cipher = Cipher(algorithms.AES(key), modes.CBC(iv),
backend=default_backend())
    decryptor = cipher.decryptor()
    decrypted_padded = decryptor.update(encrypted_data) +
decryptor.finalize()

    unpadder = padding.PKCS7(128).unpadder()
    data = unpadder.update(decrypted_padded) + unpadder.finalize()

    decrypted_file = filepath.replace('.enc', '.dec')
    with open(decrypted_file, 'wb') as f:
        f.write(data)

    print(f"File decrypted and saved as: {decrypted_file}")


# Example usage
if __name__ == "__main__":
    import argparse
    parser = argparse.ArgumentParser(description="AES-256 File Encryptor/Decryptor")
    parser.add_argument("mode", choices=["encrypt", "decrypt"], help="Mode: encrypt or decrypt")
    parser.add_argument("password", help="Password for key derivation")
    parser.add_argument("filepath", help="Path to the input file")

    args = parser.parse_args()

    if args.mode == "encrypt":
        encrypt_file(args.password, args.filepath)
    else:
        decrypt_file(args.password, args.filepath)
```