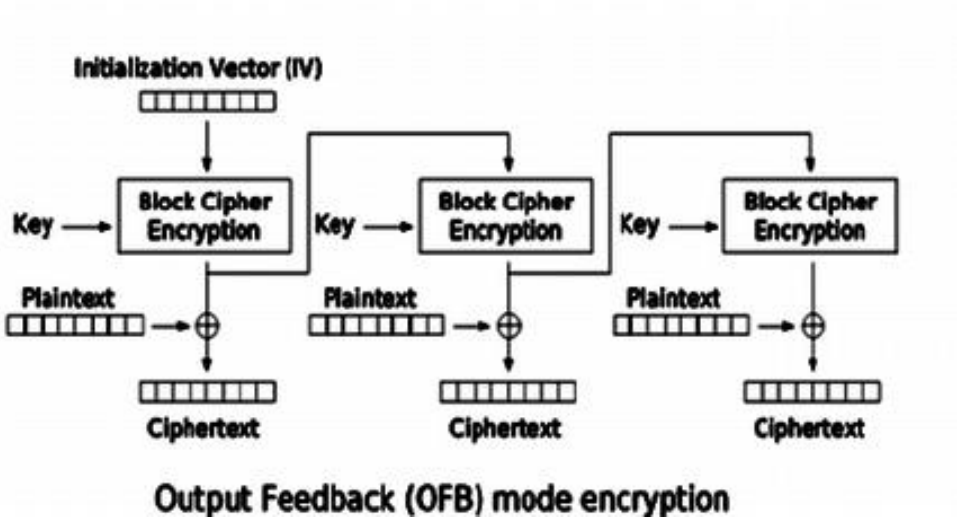
Kajetan Harla 148369

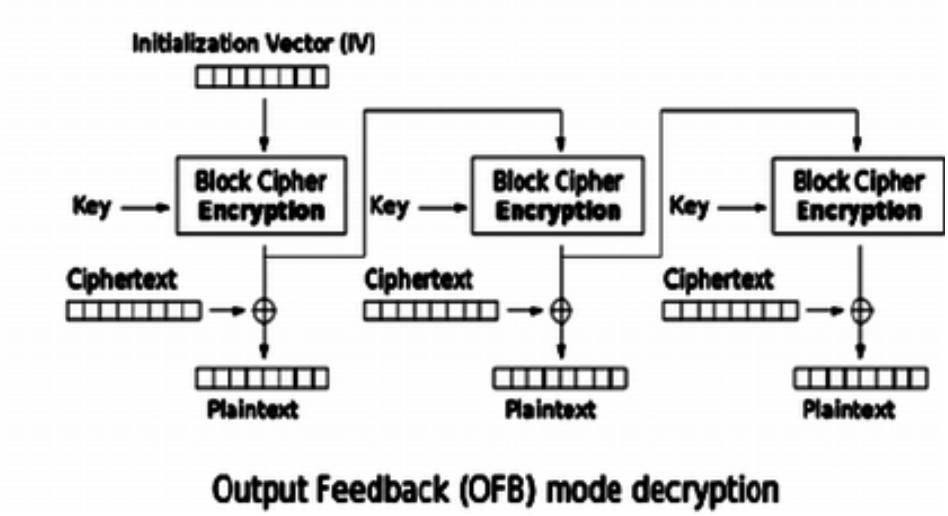
Krzysztof Surażyński 162839

Celem ćwiczenia było zaszyfrowanie i odszyfrowanie dużego pliku (o rozmiarze powyżej 2 GB) za pomocą 3DES w trybie **OFB** oraz pomiar czasu wykonywania programu.

Tryb OFB (Output Feedback Mode) to tryb pracy szyfrów blokowych, w którym blok szyfrowania działa jak generator strumienia kluczy. Każdy blok szyfrujący jest tworzony przez iteracyjne szyfrowanie wektora inicjalizacyjnego (IV) za pomocą klucza. Otrzymany wynik jest następnie XOR-owany z danymi wejściowymi w celu uzyskania szyfrogramu. OFB nie wymaga do rozszyfrowania bezpośredniego użycia odwrotnej funkcji szyfrowania – wystarczy ta sama sekwencja kluczy. Tryb ten zapobiega propagacji błędów, ale jest podatny na ataki, jeśli ten sam IV zostanie użyty wielokrotnie. Rysunki 1 i 2 zawierają schematy działania szyfrowania i deszyfrowania w trybie OFB.



Rysunek 1 Schemat szyfrowania w trybie OFB



Rysunek 2 Schemat deszyfrowania w trybie OFB

Kod programu:

```
import time
from Crypto.Cipher import DES3
from Crypto.Random import get_random_bytes

def encrypt_file(input_file, output_file):
    key = DES3.adjust_key_parity(get_random_bytes(24))
    cipher = DES3.new(key, DES3.MODE_OFB)

    with open(input_file, 'r', encoding='utf-8') as f:
        data = f.read()

    plaintext = data.encode('utf-8')

    start_time = time.time()
    print("Encryption started...")

    encrypted_msg = cipher.iv + cipher.encrypt(plaintext)

    end_time = time.time()
    total_time = end_time - start_time

    print(f"Encryption of {input_file} took {total_time:.4f} seconds.")

    with open(output_file, 'wb') as f:
        f.write(encrypted_msg)
        f.close()

    print(f"Encryption complete. Encrypted data saved to {output_file}")
    return key

def decrypt_file(input_file, output_file, key):
    with open(input_file, 'rb') as f:
        encrypted_data = f.read()
        f.close()

    iv = encrypted_data[:8]
    encrypted_msg = encrypted_data[8:]

    cipher = DES3.new(key, DES3.MODE_OFB, iv)

    start_time = time.time()
    print("Decryption started...")
```

```
    decrypted_msg = cipher.decrypt(encrypted_msg)

    end_time = time.time()
    total_time = end_time - start_time

    print(f"Decryption of {input_file} took {total_time:.4f} seconds.")

    data = decrypted_msg.decode('utf-8')

    with open(output_file, 'w', encoding='utf-8') as f:
        f.write(data)
        f.close()
    print(f"Decryption complete. Decrypted data saved to {output_file}")

used_key = encrypt_file('large_data.json', 'encrypted_data.bin')
decrypt_file('encrypted_data.bin', 'decrypted_data.json', used_key)
print(used_key)
```

Program składa się z dwóch funkcji: encrypt oraz decrypt, które odpowiednio szyfrują i odszyfrowują plik o wskazanej nazwie za pomocą 3DES w trybie OFB. Każda z funkcji zawiera również pomiar czasu wykonania operacji, obsługę plików wejściowych i wyjściowych (dane są wczytywane z pliku, przetwarzane, a następnie zapisywane w odpowiednim formacie), generowanie i przechowywanie IV (pierwszy blok zaszyfrowanego pliku). Dodatkowo, funkcja encrypt zwraca klucz wykorzystany do zaszyfrowania danych, którego można użyć do ich odszyfrowania.

Wyniki:

Szyfrowanie trwało 91.6295 sekund

Odszyfrowanie trwało 89.8344 sekund

Rozmiar pliku: 2.1 GB

Format wygenerowanego pliku: JSON

Konfiguracja sprzętowa:

- CPU: Apple M1 Max
- RAM: 32 GB

Podsumowanie i wnioski

Szyfrowanie i odszyfrowanie pliku trwało ok. 90 sekund w obu przypadkach co świadczy o podobnej złożoności obliczeniowej tych dwóch operacji. Ze względu na dużą liczbę pamięci RAM wczytanie całego wykorzystywanego pliku nie stanowiło problemu, jednak w przypadku mniejszego RAMu lub większego pliku należałoby rozważyć przetwarzanie pliku w blokach.