Sprawozdanie

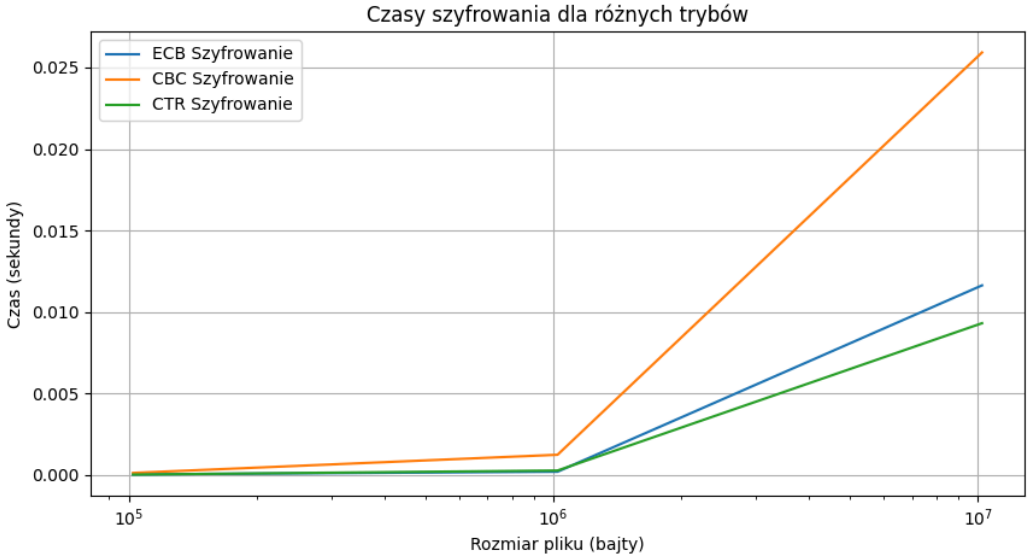
Cel zadania

- **Zadanie 1:** Zmierzyć czasy szyfrowania/deszyfrowania dla 3 rozmiarów plików (ok. 100KB, 1MB, 10MB) w 3 trybach AES (ECB, CBC, CTR) i zinterpretować wyniki.
- **Zadanie 2:** Zanalizować propagację błędów w szyfrogramach – sprawdzić, czy pojedynczy błąd skutkuje utratą całej wiadomości czy jedynie fragmentu.
- **Zadanie 3:** Zaimplementować tryb CBC przy użyciu trybu ECB.

Implementacja i wyniki

1. Pomiar czasów

- **Generacja plików:** Dla zadanych rozmiarów generowane są pliki z losowymi danymi.
- **Funkcja pomiaru:** `zmierzCzasSzyfrowaniaIDeszyfrowania` odczytuje plik, szyfruje ii mierzy czas operacji.



- **Wyniki:**
    - **CTR i ECB** – czasy prawie jednakowe
    - **CBC**- nieco wolniejszy przez wykorzystywanie dodatkowej operacji xor w każdym kroku

2. Propagacja błędów

- **Procedura:** Po szyfrowaniu wprowadzany jest błąd (zmiana jednego bajtu w środku szyfrogramu) i wykonywane jest deszyfrowanie.
- **Obserwacje:**

ECB (Electronic Codebook)

Błąd wpływa tylko na jeden blok danych. Nie występuje propagacjabłędu – inne bloki pozostają nienaruszone.

CBC (Cipher Block Chaining)

Błąd wpływa na dwa bloki: ten, w którym wystąpił (zostaje kompletnie zniekształcony), oraz kolejny. Dalsze bloki są poprawne. Występuje propagacja błędu do dwóch bloków.

OFB (Output Feedback)

Błąd wpływa tylko na jeden bajt danych. Nie występuje propagacja błędu. Tryb działa jak szyfr strumieniowy – odporny na błędy transmisji.

CFB (Cipher Feedback)

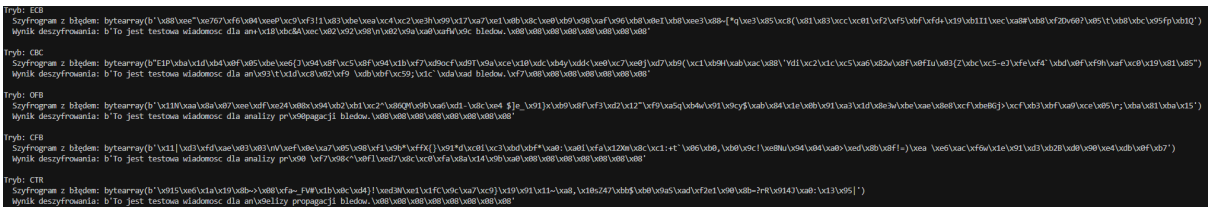
Błąd wpływa na kilka bajtów. Występuje częściowa propagacja błędu.

CTR (Counter)

Błąd wpływa tylko na jeden bajt. Nie występuje propagacja błędu. Tryb działa jak szyfr strumieniowy – odporny na błędy transmisji.

Wnioski: Tryby lokalizujące błąd (np. CTR, OFB) są bardziej odporne na pojedyncze błędy transmisji.

- **Wyniki:**



3. Implementacja trybu CBC przy użyciu ECB

- **Metoda:**
    - **Szyfrowanie:** Pierwszy blok jest XOR-owany z IV, kolejne bloki z poprzednim szyfrogramem, a następnie szyfrowane trybem ECB.
    -
    - **Deszyfrowanie:** Odszyfrowane bloki są XOR-owane z IV lub poprzednim blokiem szyfrogramu.
- **Weryfikacja:** Implementacja została sprawdzona asercją, która potwierdza, że odszyfrowany tekst odpowiada tekstowi oryginalnemu.

```python
def zaimplementujCbcUzywajacEcb(tekstJawny, klucz, iv):
    szyfrEcb = AES.new(klucz, AES.MODE_ECB)
    tekstSzyfrowany = b""
    poprzedniBlok = iv

    for i in range(0, len(tekstJawny), ROZMIAR_BLOKU):
        blok = tekstJawny[i:i + ROZMIAR_BLOKU]
        if len(blok) < ROZMIAR_BLOKU:
            blok = pad(blok, ROZMIAR_BLOKU)
        blokXor = bytes([_a ^ _b for _a, _b in zip(blok, poprzedniBlok)])
        zaszyfrowanyBlok = szyfrEcb.encrypt(blokXor)
        tekstSzyfrowany += zaszyfrowanyBlok
        poprzedniBlok = zaszyfrowanyBlok

    return tekstSzyfrowany
```

-
```python
def deszyfrujCbcUzywajacEcb(tekstSzyfrowany, klucz, iv):
    szyfrEcb = AES.new(klucz, AES.MODE_ECB)
    tekstJawny = b""
    poprzedniBlok = iv

    for i in range(0, len(tekstSzyfrowany), ROZMIAR_BLOKU):
        blok = tekstSzyfrowany[i:i + ROZMIAR_BLOKU]
        odszyfrowanyBlok = szyfrEcb.decrypt(blok)
        blokXor = bytes([_a ^ _b for _a, _b in zip(odszyfrowanyBlok, poprzedniBlok)])
        tekstJawny += blokXor
        poprzedniBlok = blok

    return unpad(tekstJawny, ROZMIAR_BLOKU)
```

```python
klucz = get_random_bytes(ROZMIAR_KLUCZA)
iv = get_random_bytes(ROZMIAR_BLOKU)
tekstJawny = b"To jest testowa wiadomosc dla implementacji trybu CBC."
tekstJawny = pad(tekstJawny, ROZMIAR_BLOKU)

tekstSzyfrowany = zaimplementujCbcUzywajacEcb(tekstJawny, klucz, iv)
odszyfrowanyTekst = deszyfrujCbcUzywajacEcb(tekstSzyfrowany, klucz, iv)

assert unpad(tekstJawny, ROZMIAR_BLOKU) == odszyfrowanyTekst, "Implementacja CBC nie powiodła się!"
print("Implementacja CBC przy użyciu trybu ECB zakończona sukcesem.")
```