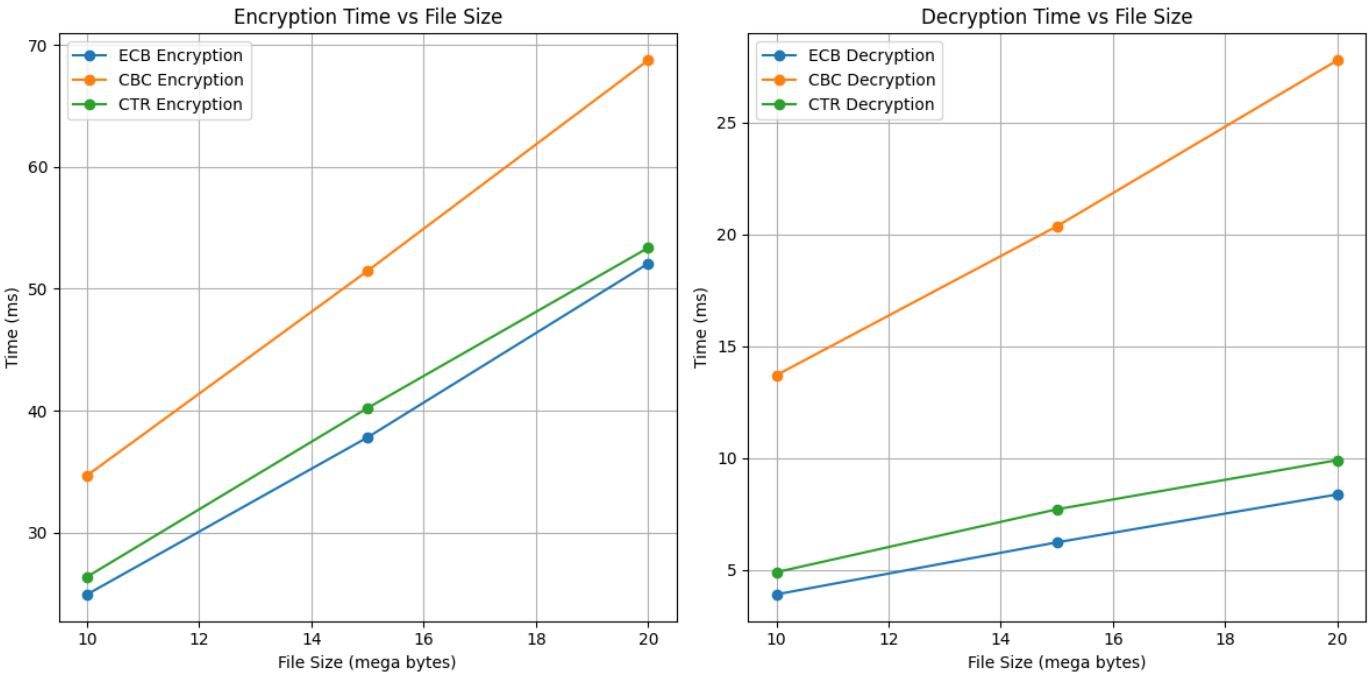
Miłosz Sobol 155905 Szyfry Blokowe

1. Czasy szyfrowania oraz deszyfrowania dla poszczególnych trybów AES

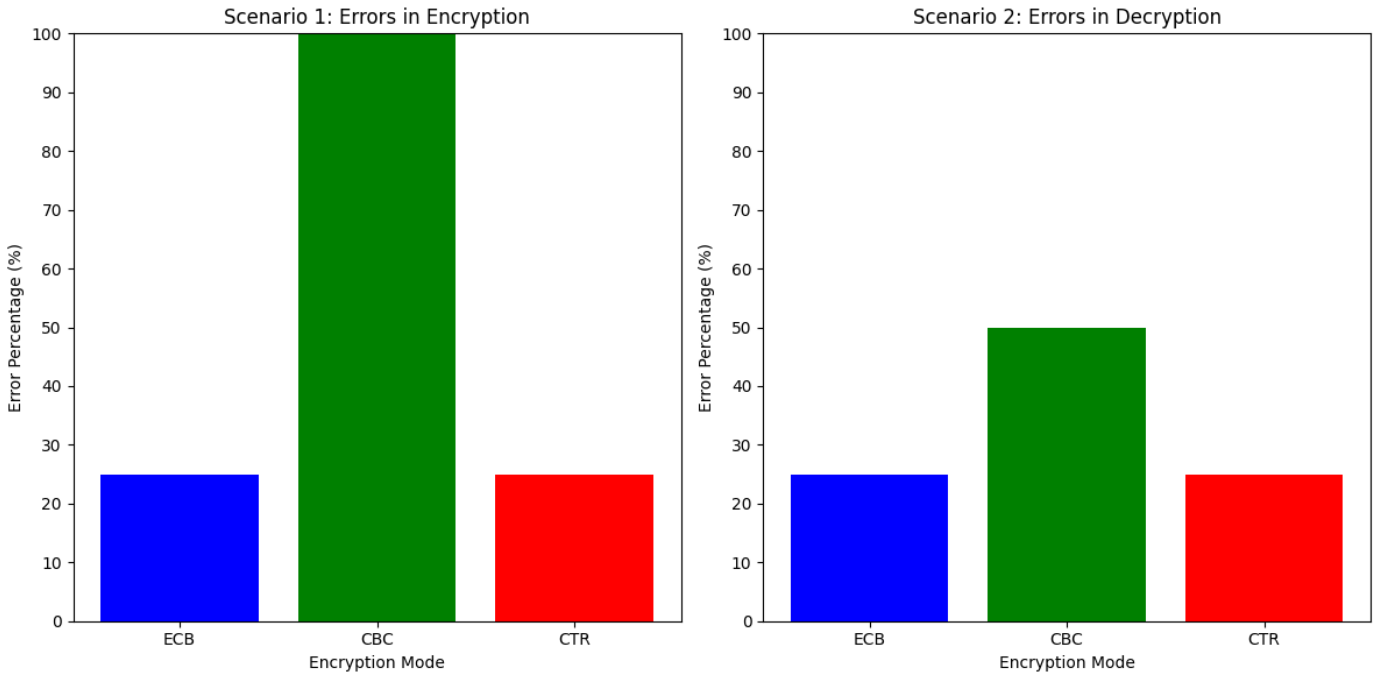


Badanie wydajnościowe zostało przeprowadzone dla plików o wielkości 10, 15 oraz 20 MB w 40 iteracjach, wartości czasowe zostały uśrednione w celu uzyskania wiarygodnych wyników. Pliki, wartości nonce oraz iv pozostały spójne na przestrzeni wszystkich iteracji.

Zdecydowanie najwolniejszym trybem działania AES jest CBC, wynika to z natury tego trybu. Każdy blok szyfrowany jest z użyciem poprzedniego zaszyfrowanego bloku, przez co równoległa implementacja tego algorytmu jest niemożliwa.

Czasy wykonywania ECB oraz CTR są bardzo zbliżone, oba te algorytmu nadają się do równoległej implementacji. W trybie ECB każdy blok danych szyfrowany jest niezależnie od innego. Dla trybu CTR generowana jest losowa duża wartość IV, następnie jest ona szyfrowana za pomocą ECB, po wykonaniu tej operacji, IV (właściwie to nasz counter) zwiększane jest o jeden i jest ponownie szyfrowany. W trybie CTR od samego początku wiemy jakie liczby przyjmie counter, to właśnie dlatego możliwa jest implementacja równoległa. Faktycznie czasy zależne są od implementacji tych trybów w bibliotece „PyCryptodome” jednak ze względu na otrzymane wyniki, możemy założyć, że tryby te zostały w jakiś sposób zrównoleglone.

2. Propagacja błędu przy szyfrowaniu i deszyfrowaniu



Otrzymane wyniki są skutkiem sposobu działania poszczególnych algorytmów.

Szyfrowana oraz deszyfrowana wiadomość składały się z czterech bloków. W przypadku pierwszego scenariusza najmniejszą propagacje błędów wykazał tryb ECB oraz CTR, dzieje się tak, ponieważ w trybach tych szyfrowanie poszczególnego bloku odbywa się niezależnie od innych bloków, tak więc błąd w pierwszym bloku nie propaguje się do następnego. W trybie CBC zaś zachodzi do propagacji, szyfrowanie każdego następnego bloku zależne jest od poprzedniego stąd błąd na samym początku skutkuje błędem w każdym następnym bloku.

W przypadku deszyfrowania dla ECB oraz CTR wyjaśnienie błędu jest analogiczne względem szyfrowania. W trybie CBC zaś dochodzi do propagacji błędu jedynie do następnego bloku, błąd w bloku pierwszym propaguje się jedynie do bloku drugiego.

3. Implementacja CBC za pomocą ECB

```python
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad

class CBC:
    def __init__(self, key, iv):
        self.key = key
        self.iv = iv
        self.block_size = 16
        self.cipher = AES.new(key, AES.MODE_ECB)

    def encrypt(self, message):
        message = pad(message, self.block_size)

        previousBlock = self.iv
        encrypted = b''

        for i in range(0, len(message), self.block_size):
            block = message[i:i + self.block_size]
            block = bytes(a ^ b for a, b in zip(block, previousBlock))
            encryptedBlock = self.cipher.encrypt(block)
            encrypted += encryptedBlock
            previousBlock = encryptedBlock

        return encrypted

    def decrypt(self, encrypted):
        previousBlock = self.iv

        decrypted = b''

        for i in range(0, len(encrypted), self.block_size):
            block = encrypted[i:i + self.block_size]
            decryptedBlock = self.cipher.decrypt(block)
            decrypted += bytes(a ^ b for a, b in zip(decryptedBlock, previousBlock))
            previousBlock = block

        return unpad(decrypted, self.block_size)
```

W CBC każdy następny szyfrowany blok jest xorowany z poprzednim blokiem, a następnie jest on szyfrowany za pomocą ECB, podobna zależność zachodzi w przypadku deszyfrowania.

```
Original: b'Czy dziala CBC?'
Encrypted: b'\xc9\x97\x08\xd8\x10\x07h\xd9\x16\xac\x83\xf6\xe0Nt~'
Decrypted: b'Czy dziala CBC?'
```