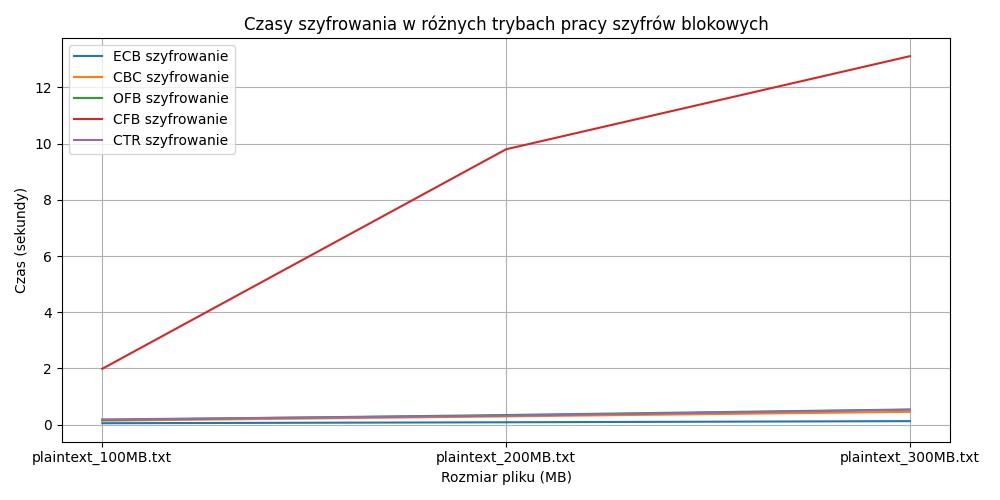
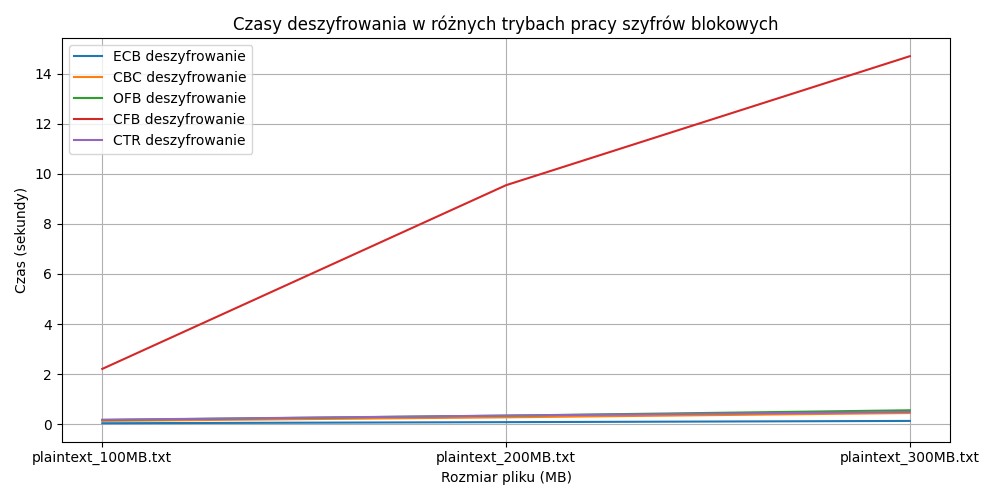
Sprawozdanie – Szyfry blokowe

Mateusz Kreczmer 151736

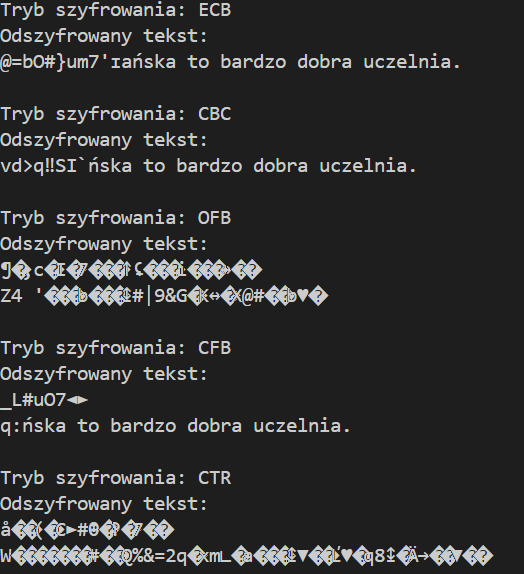
1. Przeanalizuj dostępne tryby pracy szyfrów blokowych w wybranym środowisku programowania i zmierz czasy szyfrowania i deszyfrowania dla 3 różnej wielkości plików we wszystkich 5 podstawowych trybach ECB, CBC, OFB, CFB, i CTR. Zinterpretuj otrzymane wyniki.





Szyfrowanie oraz deszyfrowanie w trybie CFB wykazuje znacznie dłuższy czas przetwarzania w porównaniu do pozostałych trybów pracy szyfrów blokowych (ECB, CBC, OFB, CTR) dla plików o rozmiarach 100 MB, 200 MB i 300 MB. Podczas gdy pozostałe tryby utrzymują czasy przetwarzania poniżej jednej sekundy dla każdej próbki danych, czas pracy trybu CFB rośnie do kilku sekund, osiągając wartości od 2 do nawet 14 sekund w zależności od rozmiaru pliku.

1. Przeanalizuj propagację błędów w wyżej wymienionych trybach pracy. Czy błąd w szyfrogramie będzie skutkował niemożnością odczytania po deszyfrowaniu całej wiadomości, fragmentu, ..? Zinterpretuj wyniki obserwacji.



Po wprowadzeniu błędu poprzez zmianę pierwszego bitu w zaszyfrowanym tekście można zauważyć, że w przypadku szyfrowań: ECB, CBC, jak i CFB jesteśmy w stanie odszyfrować większą część tekstu. Natomiast w przypadku szyfrowań OFB i CTR cały tekst jest niezdatny do odszyfrowania.

1. Zaimplementuj tryb CBC (korzystając z dostępnego w wybranym środowisku programowania trybu ECB).

W poniższej implementacji tryb CBC działa poprzez wykonywanie operacji XOR pomiędzy każdym blokiem tekstu jawnego a poprzednim zaszyfrowanym blokiem przed zaszyfrowaniem. Ten schemat zapewnia lepsze właściwości bezpieczeństwa niż sam tryb ECB, ponieważ eliminuje powtarzalność dla tych samych bloków tekstu jawnego.

1. from Crypto.Cipher import AES
2. from Crypto.Util.Padding import pad, unpad
3. from Crypto.Random import get\_random\_bytes
4. def cbc\_encrypt(plaintext, key, iv):
5. cipher = AES.new(key, AES.MODE\_ECB)
6. ciphertext = b''
7. previous\_block = iv
9. for i in range(0, len(plaintext), 16):
10. block = plaintext[i:i+16]
12. block = bytes(x ^ y for x, y in zip(block, previous\_block))
14. encrypted\_block = cipher.encrypt(block)
16. ciphertext += encrypted\_block
18. previous\_block = encrypted\_block
20. return ciphertext
21. def cbc\_decrypt(ciphertext, key, iv):
22. cipher = AES.new(key, AES.MODE\_ECB)
23. plaintext = b''
24. previous\_block = iv
26. for i in range(0, len(ciphertext), 16):
27. block = ciphertext[i:i+16]
29. decrypted\_block = cipher.decrypt(block)
31. decrypted\_block = bytes(x ^ y for x, y in zip(decrypted\_block, previous\_block))
33. plaintext += decrypted\_block
35. previous\_block = block
37. return plaintext
38. plaintext = b"Politchnika Poznanska"
39. key = get\_random\_bytes(16)
40. iv = get\_random\_bytes(16)
41. ciphertext = cbc\_encrypt(pad(plaintext, AES.block\_size), key, iv)
42. print("CBC Encrypted:", ciphertext)
43. decrypted\_text = unpad(cbc\_decrypt(ciphertext, key, iv), AES.block\_size)
44. print("CBC Decrypted:", decrypted\_text.decode())

