Laboratorium: Zaawansowane Systemy Szyfrowania			
Laboratorium nr: 1	**Prowadzący** mgr inż. Paweł Kubczak	**Data zajęć:** 21.03.2023	**Data oddania:** 02.04.2023
Nazwa ćwiczenia: Szyfr 3DES			
Wykonał: Mariusz Bartosik	**Kierunek:** Teleinformatyka	**Grupa:** 1.1	

Punkt. 1

Algorytm DES działa na blokach danych o długości 64 bitów. Przed przystąpieniem do szyfrowania dane są poddawane permutacji początkowej (IP - Initial Permutation), która zmienia kolejność bitów w bloku danych wejściowych. Następnie blok jest dzielony na dwie części o długości 32 bitów każda, po czym dane są przetwarzane przez 16 rund funkcji Feistela.

W każdej rundzie, jedna z części bloku jest poddawana funkcji F, która składa się z kilku operacji na bitach, takich jak permutacje, XORowanie, zamiana bitów i operacje na bitach przesuwania. Operacja ta jest następnie łączona z drugą częścią bloku przy użyciu operacji XOR. Po przejściu przez 16 rund blok danych jest dzielony na dwie części, po czym są one zamieniane miejscami i poddawane permutacji końcowej (FP - Final Permutation), która jest odwróceniem permutacji początkowej.

Klucz używany w algorytmie DES ma długość 56 bitów, jednak faktycznie używany jest tylko 48 bitowy podklucz. Proces generowania podkluczy polega na przesuwaniu bitów klucza oraz permutacji bitów, dzięki czemu ostatecznie powstaje 16 podkluczy o długości 48 bitów każdy. Każdy z podkluczy jest używany w jednej z rund algorytmu.

Algorytm 3DES, znany także jako Triple DES lub TDES, jest rozszerzeniem algorytmu DES, w którym operacje DES wykonuje się trzykrotnie z różnymi kluczami. Istnieją dwa tryby działania 3DES: tryb EDE (Encrypt-Decrypt-Encrypt) oraz tryb EEE (Encrypt-Encrypt-Encrypt). W trybie EDE, blok danych jest szyfrowany kluczem pierwszym, następnie deszyfrowany kluczem drugim i ponownie szyfrowany kluczem trzecim. W trybie EEE, blok danych jest szyfrowany trzema różnymi kluczami w trzech kolejnych rundach.

3DES używa klucza o długości 168 bitów, który składa się z trzech kluczy DES o długości 56 bitów każdy. Długość klucza w 3DES jest większa niż w DES, co czyni go bardziej bezpiecznym przed atakami brute force. Jednak ze względu na to, że 3DES wymaga trzykrotnie więcej czasu na szyfrowanie i deszyfrowanie danych niż DES, został zastąpiony przez bardziej zaawansowane algorytmy szyfrowania, takie jak AES (Advanced Encryption Standard).

3DES może być stosowany z różnymi trybami szyfrowania, które określają sposób, w jaki bloki danych są dzielone na mniejsze części i szyfrowane w celu zapewnienia bezpieczeństwa transmisji. Poniżej opisane są najczęściej stosowane tryby szyfrowania 3DES:

- ECB (Electronic Codebook):
  Tryb ECB jest najprostszym trybem szyfrowania, w którym bloki danych są dzielone na równe części o długości 64 bitów i każda część jest niezależnie szyfrowana przy użyciu klucza 3DES. Ten tryb szyfrowania jest stosowany wtedy, gdy bloki danych są niezależne od siebie i nie ma potrzeby uwzględniania relacji między nimi.

- CBC (Cipher Block Chaining):
  Tryb CBC jest jednym z najczęściej stosowanych trybów szyfrowania, w którym bloki danych są dzielone na części o długości 64 bitów i następnie są szyfrowane z wykorzystaniem klucza 3DES z uwzględnieniem relacji między nimi. W tym trybie szyfrowania, każdy blok danych jest łączony z poprzednim blokiem za pomocą operacji XOR i następnie szyfrowany z wykorzystaniem klucza. To połączenie bloków umożliwia odwzorowanie relacji między nimi, co utrudnia atakującym odzyskanie informacji z szyfrowanych danych.

- CFB (Cipher Feedback):
  Tryb CFB jest podobny do trybu CBC, ale zamiast łączenia poprzedniego bloku z bieżącym blokiem za pomocą operacji XOR, łączony jest fragment zaszyfrowanego tekstu z bieżącym blokiem danych. W trybie CFB, zamiast szyfrowania całego bloku danych, szyfrowany jest jedynie fragment o długości 64 bitów. Tryb CFB jest stosowany w przypadkach, gdy dane wymagają szyfrowania w czasie rzeczywistym, a nie ma potrzeby uwzględniania relacji między blokami.

- OFB (Output Feedback):
  Tryb OFB podobnie jak tryb CFB, szyfruje jedynie część bloku danych. Jednak w trybie OFB, fragment szyfrowanego tekstu jest łączony z kluczem 3DES, aby uzyskać klucz szyfrujący do zaszyfrowania bieżącego bloku danych. Tryb OFB jest stosowany w przypadkach, gdy dane wymagają szyfrowania w czasie rzeczywistym i nie ma potrzeby uwzględniania relacji między blokami.

- CTR (Counter):
  Tryb CTR jest podobny do trybu OFB, ale zamiast łączenia fragmentu szyfrowanego tekstu z kluczem 3DES, do szyfrowania bieżącego bloku danych używany jest licznik, który jest inkrementowany dla każdego bloku danych. Szyfrowanie odbywa się poprzez zaszyfrowanie licznika z kluczem 3DES, a następnie zastosowanie operacji XOR z bieżącym blokiem danych. Tryb CTR jest stosowany w przypadkach, gdy dane wymagają szyfrowania w czasie rzeczywistym i nie ma potrzeby uwzględniania relacji między blokami.

Punkt 2.

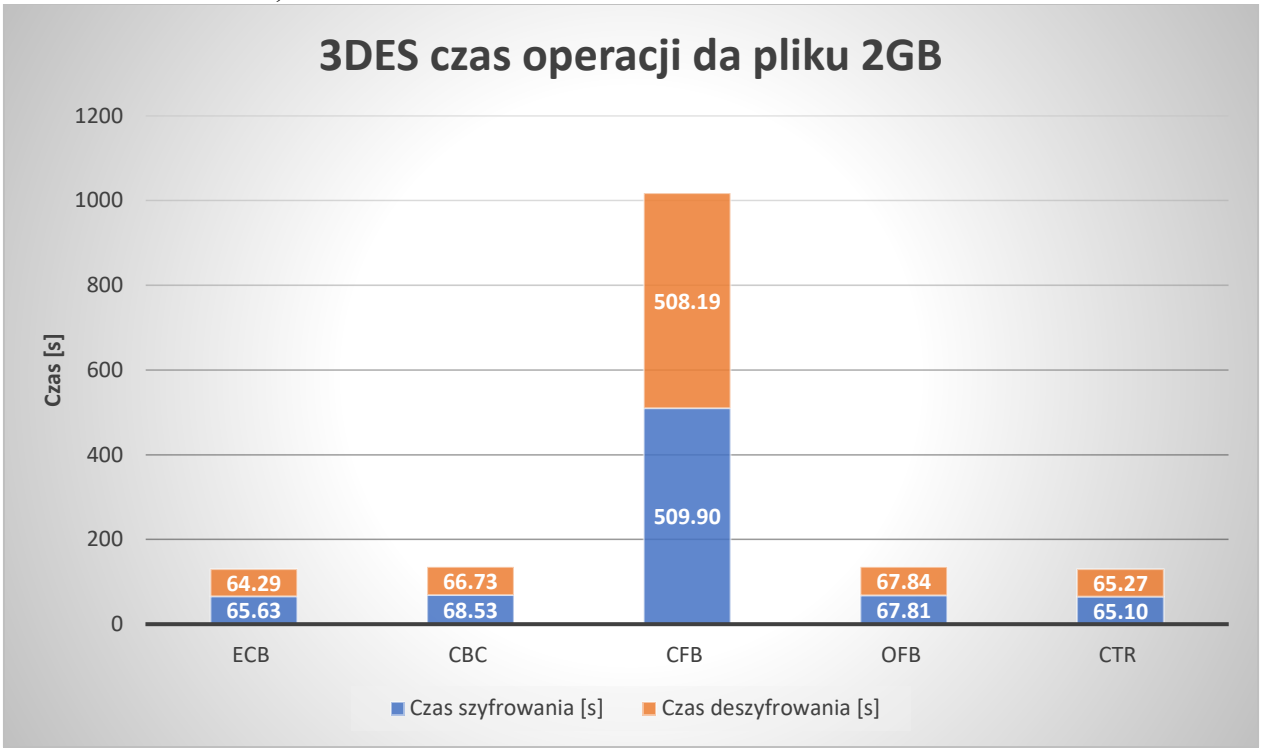
Dokonano porównania działania wybranych trybów szyfrowania poprzez zaszyfrowanie wygenerowanego pliku tekstowego o rozmiarze 2GB a następnie deszyfrując go zmierzono czas. Poniżej efekt wywołania stworzonego programu na potrzebę laboratorium przekopiowanych z konsoli. Przedstawiający czas szyfrowania oraz deszyfrowania pliku.

```
Odpowiedz programu:
MODE_ECB
Utworzono zaszyfrowany plik w sciezce: encrypted_file.bin
65.62531900405884
Utworzono zdeszyfrowany plik w sciezce: random_file1.bin
64.28747081756592
MODE_CBC
Utworzono zaszyfrowany plik w sciezce: encrypted_file.bin
68.52651953697205
Utworzono zdeszyfrowany plik w sciezce: random_file1.bin
66.73132371902466
MODE_CFB
Utworzono zaszyfrowany plik w sciezce: encrypted_file.bin
509.90143847465515
Utworzono zdeszyfrowany plik w sciezce: random_file1.bin
508.19090938568115
MODE_OFB
Utworzono zaszyfrowany plik w sciezce: encrypted_file.bin
67.81432700157166
Utworzono zdeszyfrowany plik w sciezce: random_file1.bin
67.84104251861572
MODE_CTR
Utworzono zaszyfrowany plik w sciezce: encrypted_file.bin
65.10231852531433
Utworzono zdeszyfrowany plik w sciezce: random_file1.bin
65.27120804786682
```

Kod programu zamieszczony na dole sprawozdania.

Punkt 3.
Wykorzystana została implementacja algorytmu 3DES z biblioteki „pycryptodome” ver.3.17
Pomiary przeprowadzone na jednostce:

AMD Ryzen 7 3800X 8-Core Processor 4.03 GHz
Ram 32.0 GB, 3200MHz



Rys.1 Wykres czasu szyfrowania/deszyfrowania dla różnych trybów

Analizując wyniki szyfrowania i deszyfrowania dla różnych trybów szyfrowania 3DES, można zauważyć, że czasy wykonania są zróżnicowane. Tryby szyfrowania ECB i CBC mają porównywalne czasy wykonania i są znacznie szybsze niż tryby CFB i CTR. Tryb OFB również ma zbliżony czas wykonania do trybów ECB i CBC. Czasy szyfrowania i deszyfrowania są bardzo podobne. Można to wyjaśnić tym że 3DES jest algorytmem symetrycznym co implikuję użycie takich samych operacji do deszyfrowania i do szyfrowania.

Tryb CFB charakteryzuje się najdłuższym czasem wykonania, co może wynikać z tego, że w tym trybie dane są przesyłane w mniejszych blokach, co wymaga większej liczby operacji szyfrowania i deszyfrowania. Tryb CTR również ma nieco dłuższy czas wykonania niż tryby ECB i CBC, ale wciąż jest szybszy niż tryb CFB.

Wybór trybu szyfrowania zależy od konkretnego przypadku użycia i poziomu bezpieczeństwa wymaganego przez aplikację. Tryby ECB i CBC są wystarczające w większości przypadków, ale w przypadku przetwarzania bardzo ważnych danych, należy rozważyć użycie trybów bardziej bezpiecznych, takich jak CFB czy CTR.

W trybie CFB i CTR dane są szyfrowane w sposób bardziej dynamiczny, z użyciem wektora inicjalizacji, który jest zmieniany dla każdego bloku danych. Dzięki temu dwie identyczne wartości danych nie są szyfrowane w ten sam sposób, co utrudnia atakującemu łamanie szyfru.

Bibliografia:
https://en.wikipedia.org/wiki/Triple_DES
https://csrc.nist.gov/csrc/media/publications/fips/46/3/archive/1999-10-25/documents/fips46-3.pdf
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-67r2.pdf

Kod programu.

```
from Crypto.Cipher import DES3
import enum
import time

@enum.unique
class CryptMode(enum.Enum):
    MODE_ECB = DES3.MODE_ECB
    MODE_CBC = DES3.MODE_CBC
    MODE_CFB = DES3.MODE_CFB
    MODE_OFB = DES3.MODE_OFB
    MODE_CTR = DES3.MODE_CTR

def compare_files(file1, file2):
    with open(file1, 'rb') as f1, open(file2, 'rb') as f2:
        while True:
            byte1 = f1.read(4096)
            byte2 = f2.read(4096)
            if byte1 != byte2:
                return False
            if not byte1:
                return True

def encrypt_file(input_file_path, output_file_path, key, crypt_mode):
    if(crypt_mode == CryptMode.MODE_CTR.value):
        cipher = DES3.new(key, crypt_mode, nonce=b"")
    else:
        cipher = DES3.new(key, crypt_mode)

    # Otwórz plik wejściowy i wczytaj jego zawartość
    with open(input_file_path, 'rb') as input_file:
        input_data = input_file.read()

    # Zaszyfruj dane
    encrypted_data = cipher.encrypt(input_data)
    # Zapisz zaszyfrowane dane do pliku wyjściowego
    with open(output_file_path, 'wb') as output_file:
        output_file.write(encrypted_data)
    print(f"Utworzono zaszyfrowany plik w sciezce: {output_file_path}")


def decode_file(input_file_path, output_file_path, key, crypt_mode):
    if(crypt_mode == CryptMode.MODE_CTR.value):
        cipher = DES3.new(key, crypt_mode, nonce=b"")
    else:
        cipher = DES3.new(key, crypt_mode)

    with open(input_file_path, 'rb') as input_file:
        input_data = input_file.read()

    decrypt_data = cipher.decrypt(input_data)

    with open(output_file_path, 'wb') as output_file:
```

```
        output_file.write(decrypt_data)
    print(f"Utworzono zdeszyfrowany plik w sciezce: {output_file_path}")

key = b'1234567890123456'

encodeTimes = {}
decodeTimes = {}
for cryptMode in CryptMode.__members__.values():
    print(cryptMode.name)
    stoper = time.time()
    encrypt_file('random_file.bin', 'encrypted_file.bin', key, cryptMode.value)
    encodeTimes[cryptMode.name] = abs(stoper - time.time())
    print(abs(stoper - time.time()))
    stoper = time.time()
    decode_file('encrypted_file.bin', 'random_file1.bin', key, cryptMode.value)
    decodeTimes[cryptMode.name]= abs(stoper - time.time())
    print(abs(stoper - time.time()))
```

```
import random

def generate_random_file(file_path, size_in_bytes):
    with open(file_path, 'wb') as file:
        chunk_size = 1024 * 1024  # 1 MB
        chunks = size_in_bytes // chunk_size

        for i in range(chunks):
            chunk = bytearray(random.getrandbits(8) for _ in range(chunk_size))
            file.write(chunk)

        remaining_bytes = size_in_bytes % chunk_size
        if remaining_bytes:
            chunk = bytearray(random.getrandbits(8) for _ in range(remaining_bytes))
            file.write(chunk)

    print(f"Utworzono plik o rozmiarze {size_in_bytes} bajtów w ścieżce: {file_path}")

generate_random_file('random_file.txt', 2 * 1024 * 1024 * 1024)
```