Wizualna analiza wpływu różnych trybów szyfrowania blokowego na obraz: Celem zadania jest zrozumienie, jak różne tryby szyfrowania blokowego wpływają na zaszyfrowane dane w kontekście dyfuzji i konfuzji, poprzez prostą wizualną analizę obrazów BMP po zaszyfrowaniu.

1.1 Do rozwiązania zadania wykorzystamy bibliotekę OpenSSL oraz różne polecenia linuksowe. Zadanie dotyczy szyfrowania obrazu w trybie **Electronic Code Book (ECB)**:

- **Wczytanie obrazu:** Zachowaj nagłówek pliku BMP (pierwsze 54 bajty) bez zmian (zapisz do pliku), ponieważ zawiera on metadane obrazu.
- **Przygotowanie danych do szyfrowania:** Wyodrębnij dane obrazu (piksele) zaczynając od bajtu 54 do końca pliku. Upewnij się, że dane do szyfrowania są wielokrotnością długości bloku algorytmu AES (16 bajtów). Jeśli nie są, odpowiednio je dopełnij (np. bajtami zerowymi lub za pomocą schematu paddingu PKCS#7).
- **Szyfrowanie danych:** Wygeneruj klucz szyfrujący o długości 16 bajtów (128 bitów). Możesz użyć dowolnego ciągu bajtów lub funkcji generującej losowy klucz. Użyj algorytmu AES w trybie ECB do zaszyfrowania danych obrazu. Połącz nagłówek pliku BMP z zaszyfrowanymi danymi, zapisz go do pliku `ex11_ecb_enc.bmp`.

1.2 Do rozwiązania zadania wykorzystamy bibliotekę OpenSSL oraz różne polecenia linuksowe. Zadanie dotyczy szyfrowania obrazu w trybie **CBC (Cipher Block Chaining)**:

- **Wczytanie obrazu:** Zachowaj nagłówek pliku BMP (pierwsze 54 bajty) bez zmian (zapisz do pliku), ponieważ zawiera on metadane obrazu.
- **Przygotowanie danych do szyfrowania:** Wyodrębnij dane obrazu (piksele) zaczynając od bajtu 54 do końca pliku. Upewnij się, że dane do szyfrowania są wielokrotnością długości bloku algorytmu AES (16 bajtów). Jeśli nie są, odpowiednio je dopełnij (np. bajtami zerowymi lub za pomocą schematu paddingu PKCS#7).
- **Szyfrowanie danych:** Wygeneruj klucz szyfrujący o długości 16 bajtów (128 bitów). Możesz użyć dowolnego ciągu bajtów lub funkcji generującej losowy klucz. Wygeneruj wektor inicjalizujący (IV) o długości 16 bajtów. Użyj algorytmu AES w trybie CBC do zaszyfrowania danych obrazu. Połącz nagłówek pliku BMP z zaszyfrowanymi danymi, zapisz go do pliku `ex12_cbc_enc.bmp`.

1.3 Do rozwiązania zadania wykorzystamy bibliotekę OpenSSL oraz różne polecenia linuksowe. Zadanie dotyczy szyfrowania obrazu w trybie **CFB (Cipher Feedback)**:

- **Wczytanie obrazu:** Zachowaj nagłówek pliku BMP (pierwsze 54 bajty) bez zmian (zapisz do pliku), ponieważ zawiera on metadane obrazu.
- **Przygotowanie danych do szyfrowania:** Wyodrębnij dane obrazu (piksele) zaczynając od bajtu 54 do końca pliku. Upewnij się, że dane do szyfrowania są wielokrotnością długości bloku algorytmu AES (16 bajtów). Jeśli nie są, odpowiednio je dopełnij (np. bajtami zerowymi lub za pomocą schematu paddingu PKCS#7).
- **Szyfrowanie danych:** Wygeneruj klucz szyfrujący o długości 16 bajtów (128 bitów). Możesz użyć dowolnego ciągu bajtów lub funkcji generującej losowy klucz. Wykorzystaj IV wygenerowany w zadaniu 1.2. Użyj algorytmu AES w trybie CFB do zaszyfrowania danych obrazu. Połącz nagłówek pliku BMP z zaszyfrowanymi danymi, zapisz go do pliku `ex13_cfb_enc.bmp`.

1.4 Do rozwiązania zadania wykorzystamy bibliotekę OpenSSL oraz różne polecenia linuksowe. Zadanie dotyczy szyfrowania obrazu w trybie **OFB (Output Feedback)**:

- **Wczytanie obrazu:** Zachowaj nagłówek pliku BMP (pierwsze 54 bajty) bez zmian (zapisz do pliku), ponieważ zawiera on metadane obrazu.
- **Przygotowanie danych do szyfrowania:** Wyodrębnij dane obrazu (piksele) zaczynając od bajtu 54 do końca pliku. Upewnij się, że dane do szyfrowania są wielokrotnością długości bloku algorytmu AES (16 bajtów). Jeśli nie są, odpowiednio je dopełnij (np. bajtami zerowymi lub za pomocą schematu paddingu PKCS#7).
- **Szyfrowanie danych:** Wygeneruj klucz szyfrujący o długości 16 bajtów (128 bitów). Możesz użyć dowolnego ciągu bajtów lub funkcji generującej losowy klucz. Wykorzystaj IV wygenerowany w zadaniu 1.2. Użyj algorytmu AES w trybie OFB do zaszyfrowania danych obrazu. Połącz nagłówek pliku BMP z zaszyfrowanymi danymi, zapisz go do pliku `ex14_ofb_enc.bmp`.

1.5 Do rozwiązania zadania wykorzystamy bibliotekę OpenSSL oraz różne polecenia linuksowe. Zadanie dotyczy szyfrowania obrazu w trybie **CTR (Counter)**:

- **Wczytanie obrazu:** Zachowaj nagłówek pliku BMP (pierwsze 54 bajty) bez zmian (zapisz do pliku), ponieważ zawiera on metadane obrazu.
- **Przygotowanie danych do szyfrowania:** Wyodrębnij dane obrazu (piksele) zaczynając od bajtu 54 do końca pliku. Upewnij się, że dane do szyfrowania są wielokrotnością długości bloku algorytmu AES (16 bajtów). Jeśli nie są, odpowiednio je dopełnij (np. bajtami zerowymi lub za pomocą schematu paddingu PKCS#7).
- **Szyfrowanie danych:** Wygeneruj klucz szyfrujący o długości 16 bajtów (128 bitów). Możesz użyć dowolnego ciągu bajtów lub funkcji generującej losowy klucz. Wykorzystaj IV wygenerowany w zadaniu **1.2**. Użyj algorytmu AES w trybie CTR (Counter) do zaszyfrowania danych obrazu. Połącz nagłówek pliku BMP z zaszyfrowanymi danymi, zapisz go do pliku `ex15_ctr_enc.bmp`.

1.6 Porównaj wizualnie, jak różne tryby szyfrowania wpływają na widoczność struktury oryginalnego obrazu. Zwróć szczególną uwagę na to, czy kształty i wzory z oryginalnego obrazu są nadal rozpoznawalne po zaszyfrowaniu.

1.7 Dla jednego z trybów (np. CBC): odszyfruj zaszyfrowany obraz. Upewnij się, że używasz tego samego klucza i IV (jeśli dotyczy), co podczas szyfrowania. Zapisz odszyfrowany obraz i porównaj go z oryginałem, aby potwierdzić poprawność procesu szyfrowania i odszyfrowywania.

1.8 Zmiana bajtów w plikach: zapisz do pliku `ex18.txt` wynik polecenia `uname -ra`. Dokonaj zmian wybranych bajtów w pliku. Wyświetl zawartość pliku, w jaki sposób się zmieniła?

1.9 Jak zmiany w szyfrogramie wpływają na tekst jawny? Celem zadania jest pokazanie, jak zmiany w zaszyfrowanych danych wpływają na wynik odszyfrowywania, poprzez eksperymenty z modyfikacją bajtów w zaszyfrowanych plikach obrazów BMP. Wybierz kilka bajtów w zaszyfrowanym pliku (może to być np plik `ex14_ofb_enc.bmp` z zadania **1.4**), które chcesz zmodyfikować, pamiętając, aby modyfikować bajty w sekcji danych obrazu (po nagłówku BMP), aby nie uszkodzić struktury pliku. Następnie, używając tego samego klucza i IV, które były użyte podczas szyfrowania, odszyfruj zmodyfikowany plik. Porównaj oryginalny odszyfrowany obraz z odszyfrowanym obrazem po modyfikacji.

Rozwiązania

1.1
- Do wycięcia z pliku określonej liczby bajtów możesz użyć polecenia `dd`.
- Do wyodrębnienia danych możesz użyć polecenia `dd`. Rozmiar pliku możesz sprawdzić za pomocą polecenia `stat`. Do dopełnienia możesz wykorzystać urządzenie `/dev/zero`.
- Do wygenerowania klucza możesz użyć biblioteki `OpenSSL`.

1.2
- Do wycięcia z pliku określonej liczby bajtów możesz użyć polecenia `dd`.
- Do wyodrębnienia danych możesz użyć polecenia `dd`. Rozmiar pliku możesz sprawdzić za pomocą polecenia `stat`. Do dopełnienia możesz wykorzystać urządzenie `/dev/zero`.
- Do wygenerowania klucza oraz IV możesz użyć biblioteki `OpenSSL`.

1.3
- Do wycięcia z pliku określonej liczby bajtów możesz użyć polecenia `dd`.
- Do wyodrębnienia danych możesz użyć polecenia `dd`. Rozmiar pliku możesz sprawdzić za pomocą polecenia `stat`. Do dopełnienia możesz wykorzystać urządzenie `/dev/zero`.
- Do wygenerowania klucza możesz użyć biblioteki `OpenSSL`.

1.4
- Do wycięcia z pliku określonej liczby bajtów możesz użyć polecenia `dd`.
- Do wyodrębnienia danych możesz użyć polecenia `dd`. Rozmiar pliku możesz sprawdzić za pomocą polecenia `stat`. Do dopełnienia możesz wykorzystać urządzenie `/dev/zero`.
- Do wygenerowania klucza możesz użyć biblioteki `OpenSSL`.

1.5
- Do wycięcia z pliku określonej liczby bajtów możesz użyć polecenia `dd`.
- Do wyodrębnienia danych możesz użyć polecenia `dd`. Rozmiar pliku możesz sprawdzić za pomocą polecenia `stat`. Do dopełnienia możesz wykorzystać urządzenie `/dev/zero`.
- Do wygenerowania klucza możesz użyć biblioteki `OpenSSL`.

1.6 Który z wybranych trybów nie spełnia naszych oczekiwań? Dlaczego?

1.7 Do porównania obrazów możesz wykorzystać narzędzie `diff`.

1.8 Do rozwiązania zadania możesz użyć narzędzia `xxd`. Narzędzie `xxd` pozwala na wyświetlenie pliku w formacie heksadecymalnym, co ułatwia modyfikację poszczególnych bajtów. Wykorzystując narzędzie `xxd`, kolejne kroki to: wyeksportowanie pliku `*.txt` do formatu heksadecymalnego `*.hex`, edycja pliku heksadecymalnego, zamiana pliku z powrotem na `*.txt`.

1.9 Zmiany bajtów możesz dokonać za pomocą narzędzia `xxd`.