Security: Szyfrowanie Symetryczne

Do realizacji zadania wykorzystano WSL1 na systemie Windows 2019 server.
Dystrybucja uruchomiona pod systemem WSL to Ubuntu 22.04.4 LTS
Do włączenia legacy providera w OpenSSL wykorzystano poniższy tutorial:
https://lindevs.com/enable-openssl-legacy-provider-on-ubuntu

Polecenie 1

Treść Zadania:
Stwórz plik o nazwie ptext i rozmiarze 32B z tekstem „This message is my great secret!” Wykonaj poniższe polecenie w celu zaszyfrowania tekstu z pliku ptext:
`openssl enc -e -in ptext -out ctext -aes-128-cbc -nosalt -p`
pasword: robert Wynik będzie również przedstawiał użyty klucz (key) oraz wektor inicjalizujacy (iv)
Przykładowy wynik: key= 684C851AF59965B680086B7B4896FF98 iv =4D094629F1AF902C44EA8C93E7BEA44A W sprawozdaniu umieść wywołanie komendy szyfrującej wiadomość oraz informacje, które pojawiły się na ekranie po jej wykonaniu. Dołącz plik z szyfrogramem (ctext).

Realizacja

```
root@WSL:lab4_pliki> ls
'Lab4 - szyfry symetryczne.pdf'  'Lab4 szyfry symetryczne - instrukcja.pdf'
clear   ctext_bruteforce   ecb_enc   ecb_plain_1.txt   ecb_plain_2.txt
root@WSL:lab4_pliki> cat > ptext
This message is my great secret!
^C
root@WSL:lab4_pliki> ls -lah ptext
-rwxrwxrwx 1 root root 33 Apr 13 11:50 ptext
root@WSL:lab4_pliki> openssl enc -e -in ptext -out ctext -aes-128-cbc -
nosalt -p -pass pass:robert
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
key=4007D46292298E83DA10D0763D95D513
iv =9FE0C157148D0587AA912170414CCBA6
root@WSL:lab4_pliki>
```

Polecenie 2

Treść Zadania:
Napisz i wykonaj komendy do zaszyfrowania wyżej wymienionej wiadomości ptext następującymi algorytmami: DES, 3DES, RC4. W sprawozdaniu umieść komendy szyfrujące dla każdego algorytmu

osobno. Do sprawozdania dołącz uzyskane szyfrogramy (nazwij je tak, aby ujawniały użyte algorytmy np.
`ctext_dex.txt, ctext_3des.txt, ctext_rc4.txt).`

Realizacja

```
root@WSL:lab4_pliki> ls
'Lab4 - szyfry symetryczne.pdf'  'Lab4 szyfry symetryczne - instrukcja.pdf'
clear   ctext   ctext_bruteforce   ecb_enc   ecb_plain_1.txt
ecb_plain_2.txt   ptext
root@WSL:lab4_pliki> openssl enc -e -des-cbc -in ptext -out ctext_des.txt -
nosalt -pass pass:robert
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
root@WSL:lab4_pliki> openssl enc -e -des-ede3-cbc -in ptext -out
ctext_3des.txt -nosalt -pass pass:robert
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
root@WSL:lab4_pliki> openssl enc -e -rc4 -in ptext -out ctext_rc4.txt -
nosalt -pass pass:robert
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
root@WSL:lab4_pliki> ls -lah ctext_*.txt
-rwxrwxrwx 1 root root 40 Apr 13 12:26 ctext_3des.txt
-rwxrwxrwx 1 root root 40 Apr 13 12:26 ctext_des.txt
-rwxrwxrwx 1 root root 33 Apr 13 12:26 ctext_rc4.txt
root@WSL:lab4_pliki>
```

Odpowiedzi na pytania

Jaka jest długość kluczy dla poszczególnych algorytmów ? Odpowiedź uzasadnij wskazując odpowiednie fragmenty komend szyfrujących oraz opisy algorytmów DES, 3DES, RC4.

- AES-128-CBC: Używany algorytm AES-128 posiada klucz o długości 128 bitów (16 bajtów). W naszym poleceniu opcja -aes-128-cbc informuje, że szyfrowanie odbywa się przy użyciu 128-bitowego klucza.
- DES: Algorytm DES wykorzystuje klucz o nominalnej długości 64 bity (8 bajtów). Jednakże jeden bajt na każdy 8-bitowy segment jest używany do parzystości, co daje efektywną długość klucza równą 56 bitom. W komendzie użyto opcji -des, co implikuje taki format klucza.
- 3DES (Triple DES): W trybie 3DES, używanym tutaj poleceniem -des-ede3, zazwyczaj wykorzystuje się trzy klucze DES, co daje łącznie 192 bity (24 bajty). Jednak ze względu na algorytmowe ograniczenia, efektywna długość klucza wynosi 168 bitów (ponieważ w każdym kluczu 8 bitów to bity parzystości). Uzasadnienie: Komenda -des-ede3 wskazuje na użycie trzykrotnego DES.
- RC4: RC4 to strumieniowy algorytm szyfrowania, w którym długość klucza może być zmienna.

W implementacji OpenSSL domyślnie stosowany jest klucz o długości 128 bitów (16 bajtów), chyba że jawnie określimy inną długość. Uzasadnienie: Komenda -rc4 nie precyzuje rozmiaru klucza, dlatego opiera się na ustawieniach domyślnych.

Wykonaj szyfrowanie algorytmami DES, 3DEC i RC4 wiadomości ptext, gdy jest ona krótsza i dłuższa niż 32 bajty np. wynosi 26 i 37 bajtów. Odpowiedz jaka jest długość szyfrogramów w wyżej wymienionych przypadkach. Wyjaśnij dlaczego uzyskano takie właśnie wyniki.

```
root@WSL:lab4_pliki> ls
'Lab4 - szyfry symetryczne.pdf'              clear   ctext_3des.txt
ctext_des.txt   ecb_enc           ecb_plain_2.txt
'Lab4 szyfry symetryczne - instrukcja.pdf'   ctext   ctext_bruteforce
ctext_rc4.txt   ecb_plain_1.txt   ptext
root@WSL:lab4_pliki> cat ptext
This message is my great secret!
root@WSL:lab4_pliki> cp ptext ptext_short
root@WSL:lab4_pliki> cp ptext ptext_long
root@WSL:lab4_pliki> cat > ptext_short
This message is my great
^C
root@WSL:lab4_pliki> cat > ptext_long
This message is my great secret!!!!!!!!!!!!
^C
root@WSL:lab4_pliki> ls -lah ptext_*
-rwxrwxrwx 1 root root 45 Apr 13 12:37 ptext_long
-rwxrwxrwx 1 root root 25 Apr 13 12:36 ptext_short
root@WSL:lab4_pliki> nano ptext_long
root@WSL:lab4_pliki> nano ptext_short
root@WSL:lab4_pliki> ls -lah ptext_*
-rwxrwxrwx 1 root root 37 Apr 13 12:38 ptext_long
-rwxrwxrwx 1 root root 26 Apr 13 12:38 ptext_short
root@WSL:lab4_pliki> openssl enc -e -des-cbc -in ptext_long -out
ctext_des_long.txt -nosalt -pass pass:robert
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
root@WSL:lab4_pliki> openssl enc -e -des-ede3-cbc -in ptext_long -out
ctext_3des_long.txt -nosalt -pass pass:robert
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
root@WSL:lab4_pliki> openssl enc -e -rc4 -in ptext_long -out
ctext_rc4_long.txt -nosalt -pass pass:robert
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
root@WSL:lab4_pliki> openssl enc -e -des-cbc -in ptext_short -out
ctext_3des_short.txt -nosalt -pass pass:robert
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
root@WSL:lab4_pliki> openssl enc -e -des-ede3-cbc -in ptext_short -out
```

```
ctext_3des_short.txt -nosalt -pass pass:robert
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
root@WSL:lab4_pliki> openssl enc -e -rc4 -in ptext_short -out
ctext_rc4_short.txt -nosalt -pass pass:robert
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
root@WSL:lab4_pliki> ls -lah ctext_*_*.txt
\-rwxrwxrwx 1 root root 40 Apr 13 12:42 ctext_3des_long.txt
-rwxrwxrwx 1 root root 32 Apr 13 12:43 ctext_3des_short.txt
-rwxrwxrwx 1 root root 40 Apr 13 12:41 ctext_des_long.txt
-rwxrwxrwx 1 root root 37 Apr 13 12:43 ctext_rc4_long.txt
-rwxrwxrwx 1 root root 26 Apr 13 12:43 ctext_rc4_short.txt
root@WSL:lab4_pliki>
```

- DES i 3DES (algorytmy blokowe): Te algorytmy działają na blokach o rozmiarze 8 bajtów. Gdy wiadomość ma 26 bajtów, zostanie uzupełniona (padding) do najbliższej wielokrotności 8 – czyli do 32 bajtów (ponieważ nawet gdy długość wiadomości jest wielokrotnością bloku, stosowany jest mechanizm paddingu PKCS7). Dla wiadomości o długości 37 bajtów, padding uzupełni do 40 bajtów.

- RC4 (algorytm strumieniowy): Szyfrowanie strumieniowe nie wprowadza dodatkowego paddingu – szyfrogram ma dokładnie taką samą długość jak wiadomość jawna. Dla 26 bajtów otrzymamy 26-bajtowy szyfrogram. Dla 37 bajtów – 37 bajtowy szyfrogram.

Polecenie 3

Treść Zadania:
Co się stanie gdy użyjemy poniższego polecenia z takimi samymi wartościami klucza (key) oraz wektora inicjalizującego (iv) jak te zwrócone w wyniku wykonania polecenia 1.
`openssl enc -e -in ptext -out ctext_aes_128_cbc -K key -iv iv -aes-128-cbc -nosalt`
W sprawozdaniu umieść wynik wywołania podanej komendy oraz uzasadnij odpowiedź. Do sprawozdania dołącz otrzymany szyfrogram.

Realizacja

```
root@WSL:lab4_pliki> openssl enc -e -in ptext -out ctext_aes_128_cbc -K
684C851AF59965B680086B7B4896FF98 -iv 4D094629F1AF902C44EA8C93E7BEA44A  -
aes-128-cbc -nosalt
root@WSL:lab4_pliki> ls -lah ctext_aes_128_cbc ctext
-rwxrwxrwx 1 root root 48 Apr 13 11:50 ctext
-rwxrwxrwx 1 root root 48 Apr 13 12:47 ctext_aes_128_cbc
```

```
root@WSL:lab4_pliki>
```

Pliki mają ten sam rozmiar więc powinny być identyczne

Polecenie 4

Treść zadania:
Napisz odpowiednie polecenia do odszyfrowania wiadomości zaszyfrowanych w poleceniu 2, a następnie użyj ich tzn. odszyfruj wiadomości zaszyfrowane w poleceniu 2. Odszyfrowane wiadomości zapisz w plikach
`dtext_nazwa_algorytmu np. dtext_des, dtext_3des, dtext_rc4`
Sprawdź zwartość plików `dtext_*` i potwierdź że odszyfrowanie przebiegło pomyślnie tzn. porównaj zawartość plików ptext i `dtext_*`. Podpowiedź: użyj tych samych kluczy i wektorów inicjalizujących (jeżeli są wymagane), które były używane podczas szyfrowania wiadomości. W sprawozdaniu umieść komendy użyte do odszyfrowania, oraz porównanie zawartości plików ptext i `dtext_*`.

Realizacja

```
root@WSL:lab4_pliki> openssl enc -d -in ctext_des.txt -out dtext_des_out.txt
-des -nosalt -pass pass:robert
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
root@WSL:lab4_pliki> openssl enc -d -in ctext_3des.txt -out dtext_3des.txt -
des-ede3 -nosalt -pass pass:robert # miejsce w którym zepsułem plik gdyż
nadpisałem plik podczas jego rozszyfrowywania
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
bad decrypt
4007235C747F0000:error:1C800064:Provider routines:ossl_cipher_unpadblock:bad
decrypt:../providers/implementations/ciphers/ciphercommon_block.c:124:
root@WSL:lab4_pliki> openssl enc -d -in ctext_3des.txt -out
dtext_3des_out.txt -des-ede3 -nosalt -pass pass:robert
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
bad decrypt
4007B509A77F0000:error:1C800064:Provider routines:ossl_cipher_unpadblock:bad
decrypt:../providers/implementations/ciphers/ciphercommon_block.c:124:
root@WSL:lab4_pliki> openssl enc -d -in ctext_rc4.txt -out dtext_rc4_out.txt
-rc4 -nosalt -pass pass:robert
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
root@WSL:lab4_pliki> ls -lah *_out.txt
-rwxrwxrwx 1 root root 32 Apr 13 12:53 dtext_3des_out.txt
-rwxrwxrwx 1 root root 33 Apr 13 12:53 dtext_des_out.txt
-rwxrwxrwx 1 root root 33 Apr 13 12:54 dtext_rc4_out.txt
root@WSL:lab4_pliki> cat dtext_*_out.txt
```

```
<Tutaj były brzydkie znaki> # tutaj jest uszkodzony plik
This message is my great secret!
This message is my great secret!
root@WSL:lab4_pliki> xterm-256color
```

Wszystko zadziałało poprawnie gdyby nie to że omyłkowo sobie zepsułem jeden plik literówką w poleceniu tak to wszystko powinno działać poprawnie ;)

Odpowiedzi na pytania

Jaki jest efekt wykonania deszyfrowania z innymi kluczami ? Wykonaj próbę i zamieść w sprawozdaniu wyniki. Odpowiedź uzasadnij.

```
root@WSL:lab4_pliki> openssl enc -d -in ctext_rc4.txt -out
dtext_rc4_out_dif_key.txt -rc4 -nosalt -pass pass:key123
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
root@WSL:lab4_pliki> cat dtext_*_key.txt
<tutaj pojawiają się brzydkie znakie które mają problem z rednerowaniem się
w LaTeXu>
root@WSL:lab4_pliki>
```

- Jeżeli przy odszyfrowaniu użyjemy innego klucza niż ten, który był użyty do szyfrowania, uzyskamy:
- Wynik w postaci nieczytelnego ciągu znaków (głupkowaty tekst lub zbitek symboli),
- W niektórych przypadkach może pojawić się komunikat błędu lub ostrzeżenie dotyczące paddingu.
- Uzasadnienie: Klucz jest podstawowym parametrem determinującym, jak blok danych zostanie przetworzony. Nieprawidłowy klucz powoduje całkowite zniekształcenie odszyfrowanego tekstu, gdyż mechanizm deszyfrowania nie jest w stanie odtworzyć oryginalnych bloków danych.

Jaki jest efekt wykonania deszyfrowania z innymi wektorami inicjalizującymi ? Wykonaj próbę i zamieść w sprawozdaniu wyniki. Odpowiedź uzasadnij.

- Podobnie jak w przypadku niewłaściwego klucza, zastosowanie błędnego wektora inicjalizującego (IV) podczas odszyfrowywania – w szczególności w trybach operacji typu CBC – prowadzi do:
- Niepoprawnego odszyfrowania pierwszego bloku (lub pierwszych kilku bloków) danych,
- Możliwości częściowego odzyskania danych w kolejnych blokach (z uwagi na sprzężenie bloków), ale końcowy efekt będzie nieczytelny.
- Uzasadnienie: IV wpływa na sposób, w jaki pierwszy blok danych jest przetwarzany. Nawet przy poprawnym kluczu, błędny IV spowoduje błąd w odszyfrowaniu, co skutkuje zmianą treści pierwszego bloku, a błąd ten może wpływać na kolejne bloki, w zależności od mechanizmu sprzężenia.

Polecenie 5

Wiedząc, że szyfrogram `ctext_bruteforce` został stworzony za pomocą poniższego polecenia: `openssl enc -e -in ptext -out ctext_bruteforce -K krotki_klucz -rc4` gdzie `krótki_klucz` oznacza klucz o długości 4 bitów (jeden znak hexadecymalny) proszę dokonać odszyfrowania tego szyfrogramu i podać jaką treść posiadała oryginalna wiadomość. Należy wykonać 150 prób za każdy razem zapisując wyniki.

Realizacja

```
root@WSL:lab4_pliki> openssl enc -e -in ptext -out ctext_bruteforce -K F -
rc4
hex string is too short, padding with zero bytes to length
root@WSL:lab4_pliki> ls -lah ptext ctext_bruteforce
-rwxrwxrwx 1 root root 33 Apr 13 13:04 ctext_bruteforce
-rwxrwxrwx 1 root root 33 Apr 13 11:50 ptext
root@WSL:lab4_pliki> openssl enc -e -in ptext -out ctext_bruteforce -K F -
rc4
hex string is too short, padding with zero bytes to length
root@WSL:lab4_pliki> ls -lah ptext ctext_bruteforce
-rwxrwxrwx 1 root root 33 Apr 13 13:04 ctext_bruteforce
-rwxrwxrwx 1 root root 33 Apr 13 11:50 ptext
root@WSL:lab4_pliki> ^C
root@WSL:lab4_pliki> cat > bruteforce.sh
#!/bin/bash

for key in {0..15}; do
    # Konwertuj klucz na format hex (jedna cyfra może być zapisana z zerem
wiodącym, np. 0 -> 0)
    hex_key=$(printf "%X" $key)
    # Odszyfrowanie przy użyciu RC4 z danym kluczem
    openssl enc -d -in ctext_bruteforce -out dtext_bruteforce_$hex_key -rc4
-K $hex_key
    echo "Próba z kluczem: $hex_key" >> brute_force_log.txt
    cat dtext_bruteforce_$hex_key >> brute_force_log.txt
    echo -e "\n---\n" >> brute_force_log.txt
done

^C
root@WSL:lab4_pliki> ./bruteforce.sh
hex string is too short, padding with zero bytes to length
...
hex string is too short, padding with zero bytes to length
root@WSL:lab4_pliki> cat brute_force_log.txt | tail

Próba z kluczem: E
<brzydkie znaki>
```

```
---

Próba z kluczem: F
This message is my great secret!

---
```

Odpowiedzi na pytania

Ile prób spośród 150 było udanych. Wyjaśnij dlaczego. W sprawozdaniu zapisz odpowiedź z uzasadnieniem. Do sprawozdania dołącz plik dokumentujący wykonane próby (150).

Jeżeli wiemy że klucz ma długość 4bitów to nie ma potrzeby wykonywania 150 prób wystarczy ich 16 (tyle ile wartości jest w systemie heksadecymalnym). Wybrałem klucz F więc ostatnio próba była celna.
Uzasadnienie:
W przypadku klucza o długości 4 bitów możliwych jest tylko 16 unikalnych kluczy (0–F). Niezależnie od liczby powtórzonych prób (150 prób) tylko jedna próba może być udana, ponieważ tylko jeden z 16 kluczy jest poprawny. Powtórzenie tych 16 prób wielokrotnie nie zwiększa liczby poprawnych odszyfrowań – uzyskamy 150 odszyfrowań, z których tylko 16 jest unikalnych, a tylko 1 jest prawidłowym odszyfrowaniem oryginalnej wiadomości.

Polecenie 6

W tym ćwiczeniu należy użyć plików: `ecb_enc, ecb_plain_1, ecb_plain_2` Wiedząc, że plik `ecb_enc` jest szyfrogramem jednego z plików jawnych: `ecb_plain_1 lub ecb_plain_2`, które zostały zaszyfrowane algorytmem DES w trybie ecb, proszę określić który z plików jest tekstem jawnym odpowiadającym szyfrogramowi zawartemu w pliku `ecb_enc`.

Realizacja

```
root@WSL:lab4_pliki> hexdump -C ecb_enc > ecb_enc_hex.txt
root@WSL:lab4_pliki> hexdump -C ecb_plain_1 > ecb_plain_1_hex.txt
hexdump: ecb_plain_1: No such file or directory
hexdump: all input file arguments failed
root@WSL:lab4_pliki> ls
'Lab4 - szyfry symetryczne.pdf'              clear
ctext_3des_short.txt   ctext_des_long.txt    dtext_3des.txt
dtext_rc4_out_dif_key.txt   ecb_plain_1_hex.txt   ptext_short
'Lab4 szyfry symetryczne - instrukcja.pdf'   ctext
ctext_aes_128_cbc      ctext_rc4.txt         dtext_3des_out.txt   ecb_enc
ecb_plain_2.txt
```

```
brute_force_log.txt                          ctext_3des.txt
ctext_bruteforce       ctext_rc4_long.txt    dtext_des_out.txt
ecb_enc_hex.txt             ptext
bruteforce.sh                                ctext_3des_long.txt
ctext_des.txt          ctext_rc4_short.txt   dtext_rc4_out.txt
ecb_plain_1.txt             ptext_long
root@WSL:lab4_pliki> hexdump -C ecb_plain_1.txt > ecb_plain_1_hex.txt
root@WSL:lab4_pliki> hexdump -C ecb_plain_2.txt > ecb_plain_2_hex.txt
root@WSL:lab4_pliki> # Przykładowo, wyodrębnij pierwszy blok 8-bajtowy (16
znaków hex) z każdego pliku:
root@WSL:lab4_pliki> dd if=ecb_enc bs=8 count=1 2>/dev/null | hexdump -C
00000000  05 14 e5 50 c2 e6 5a 09                           |...P..Z.|
00000008
root@WSL:lab4_pliki> dd if=ecb_plain_1 bs=8 count=1 2>/dev/null | hexdump -C
root@WSL:lab4_pliki> dd if=ecb_plain_1.txt bs=8 count=1 2>/dev/null |
hexdump -C
00000000  4d 6f 6a 65 20 69 6e 66                           |Moje inf|
00000008
root@WSL:lab4_pliki> dd if=ecb_plain_2.txt bs=8 count=1 2>/dev/null |
hexdump -C
00000000  54 6f 20 6a 65 73 74 20                           |To jest |
00000008
root@WSL:lab4_pliki>
```

Odpowiedzi na pytania

Czy na podstawie tylko analizy plików ecb-enc, ecb-plain-1, ecb-plain-2 można powiedzieć, która wiadomość została zaszyfrowana ? Dlaczego, wyjaśnij? Odpowiedź z uzasadnieniem zapisz w sprawozdaniu.

- Na podstawie tylko analizy plików `ecb_enc, ecb_plain_1 i ecb_plain_2` nie możemy jednoznacznie stwierdzić, który plik jawny został użyty do zaszyfrowania.
- Uzasadnienie: Brak powtórzeń bloków: W trybie ECB identyczne bloki w tekście jawnym powodują, że szyfrogramy będą zawierały te same bloki w tych samych miejscach. Jednak w tym przypadku pierwszy blok w szyfrogramie `(ecb_enc)` różni się od pierwszych bloków w obu plikach jawnych `(ecb_plain_1 i ecb_plain_2)`, co sugeruje, że szyfrogram może nie pochodzić od żadnego z tych dwóch plików na podstawie tylko tej analizy. Brak powtarzalności bloków: W przypadku trybu ECB, powtarzalność bloków w tekście jawnym jest kluczowym wskaźnikiem, jednak w tej analizie nie występują żadne powtarzające się bloki. Oznacza to, że sama analiza heksadecymalna pierwszych bloków nie wystarcza do jednoznacznego wskazania, który plik jawny został zaszyfrowany.
- Dalsze kroki Aby dokładniej określić, który plik jawny został zaszyfrowany, należałoby kontynuować analizę na poziomie kolejnych bloków w szyfrogramie i plikach jawnych. W szczególności, jeśli szyfrogram zawiera powtarzające się fragmenty, może to wskazywać na powtarzające się bloki w jednym z plików jawnych. Jednak na podstawie obecnej analizy nie możemy jednoznacznie wskazać, który plik został użyty.