Security: Symmetric encryption

WSL1 on Windows 2019 server was used to complete the task.
The distribution running under WSL is Ubuntu 22.04.4 LTS
The following tutorial was used to enable the legacy provider in OpenSSL:
https://lindevs.com/enable-openssl-legacy-provider-on-ubuntu

Command 1

Task content:
Create a file named ptext and size 32B with the text „This message is my great secret!”. Run the following command to encrypt the text from the ptext file:
`openssl enc -e -in ptext -out ctext -aes-128-cbc -nosalt -p`
pasword: robert The result will also show the key used (key) and the initialisation vector (iv) Example result: key= 684C851AF59965B680086B7B4896FF98 iv =4D094629F1AF902C44EA8C93E7BEA44A In the report, include the command call that encrypted the message and the information that appeared on the screen after it was executed. Include the ciphertext file.

Implementation

```
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>
```

Command 2

Task content:
Write and execute commands to encrypt the above ptext message with the following algorithms: DES, 3DES, RC4. Include the encryption commands for each algorithm separately in the report. Include the resulting ciphertexts in the report (name them so that they reveal the algorithms used e.g.

`ctext_dex.txt, ctext_3des.txt, ctext_rc4.txt).`

Implementation

```
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>
```

Answers to questions

What is the key length for the different algorithms ? Justify your answer by pointing to the relevant parts of the encryption commands and descriptions of the DES, 3DES, RC4 algorithms.

- AES-128-CBC: The AES-128 algorithm used has a key length of 128 bits (16 bytes). In our command, the -aes-128-cbc option indicates that encryption is performed using a 128-bit key.
- DES: The DES algorithm uses a key with a nominal length of 64 bits (8 bytes). However, one byte for each 8-bit segment is used for parity, giving an effective key length of 56 bits. The command uses the -des option, which implies this key format.
- 3DES (Triple DES): The 3DES mode, used here with the -des-ede3 command, typically uses three DES keys, giving a total of 192 bits (24 bytes). However, due to algorithmic limitations, the effective key length is 168 bits (as 8 bits in each key are parity bits). Justification: The -des-ede3 command indicates the use of triple DES.
- RC4: RC4 is a stream encryption algorithm in which the key length can be variable. The OpenSSL implementation uses a key length of 128 bits (16 bytes) by default, unless we explicitly specify a different length. Justification: The -rc4 command does not specify the key size, so it relies on the default settings.

Perform encryption with the DES, 3DEC and RC4 algorithms on a ptext message when it is shorter and longer than 32 bytes, e.g. it is 26 and 37 bytes. Answer what is the length of the ciphertexts in the above cases. Explain why these results were obtained.

```
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 and 3DES (block algorithms): These algorithms operate on block sizes of 8 bytes. When a message is 26 bytes long, it will be padded (padding) to the nearest multiple of 8 - that is, to 32 bytes (because even when the message length is a multiple of a block, the PKCS7 padding mechanism is used). For a message with a length of 37 bytes, padding will complete to 40 bytes.

- RC4 (stream algorithm): Stream encryption does not introduce additional padding - the ciphertext is exactly the same length as the plaintext message. For 26 bytes we get a 26-byte ciphertext. For 37 bytes - a 37-byte ciphertext.

Command 3

Task content:
What happens when we use the following command with the same key (key) and initialisation vector (iv) values as those returned by command 1.
`openssl enc -e -in ptext -out ctext_aes_128_cbc -K key -iv iv -aes-128-cbc -nosalt`
In your report, include the result of calling the given command and justify your answer. Attach the resulting ciphertext to the report.

Implementation

```
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>
```

The files are the same size so they should be identical

Command 4

Task content:
Write the appropriate commands to decrypt the messages encrypted in command 2, and then use them, i.e. decrypt the messages encrypted in command 2. Save the decrypted messages in the files `dtext_algorithm name e.g. dtext_des, dtext_3des, dtext_rc4.`
Check the contents of the files `dtext_*` and confirm that the decryption was successful, i.e. compare the contents of the ptext and `dtext_*`. Hint: use the same keys and initialisation vectors (if required) that were used when encrypting the message. In the report, include the commands used to decrypt, and a comparison of the contents of the ptext and `dtext_*`.

Implementation

```
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
```

Everything worked fine if it wasn't for the fact that I mistakenly messed up one file with a typo in the command so everything should work fine ;)

Answers to questions

What is the effect of performing decryption with different keys ? Perform the test and include the results in the report. Justify your answer.

```
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>
```

- If, in decryption, we use a different key from the one used for encryption, we obtain:
- The result in the form of an unreadable string (silly text or a cluster of symbols),
- In some cases an error message or padding warning may appear.
- Justification: The key is a fundamental parameter that determines how the data block will be processed. An incorrect key results in a complete distortion of the decrypted text, as the decryption mechanism is unable to reconstruct the original data blocks.

What is the effect of performing decryption with different initialisation vectors ? Perform the test and include the results in the report. Justify your answer.

- As in the case of an incorrect key, the use of an incorrect initialisation vector (IV) during decryption - particularly in CBC-type modes of operation - leads to:
- Incorrect decryption of the first block (or first few blocks) of data,
- Possible partial recovery of data in subsequent blocks (due to block coupling), but the final result will be unreadable.
- Rationale: IV affects the way the first block of data is processed. Even with a correct key, an incorrect IV will cause a decryption error, resulting in a change in the content of the first block, and this error may affect subsequent blocks, depending on the coupling mechanism.

Command 5

Knowing that the ciphertext `ctext_bruteforce` was created using the following command: `openssl enc -e -in ptext -out ctext_bruteforce -K tuples_key -rc4` where `short_key` denotes a key of length 4 bits (one hexadecimal character) please decrypt this ciphertext and state what the content of the original message was. Make 150 attempts each time recording the results.

Implementation

```
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!

---
```

Answers to questions

How many attempts out of 150 were successful. Explain why. In the report, record your answer with reasons. Attach to the report a file documenting the attempts made (150).

If we know that the key has a length of 4 bits then there is no need to make 150 attempts, 16 attempts are enough (the number of values in the hexadecimal system). I chose the F key so the last attempt was accurate.
Justification:
For a key of length 4 bits, only 16 unique keys (0-F) are possible. Regardless of the number of repeated attempts (150 attempts), only one attempt can be successful because only one of the 16 keys is correct. Repeating these 16 attempts multiple times does not increase the number of correct decryptions - we will get 150 decryptions, of which only 16 are unique and only 1 is a correct decryption of the original message.

Command 6

Use the files in this exercise: `ecb_enc, ecb_plain_1, ecb_plain_2.` Knowing that the file `ecb_enc` is the ciphertext of one of the plaintext files: `ecb_plain_1 or ecb_plain_2`, which were encrypted with the DES algorithm in ecb mode, please determine which file is the plaintext corresponding to the ciphertext contained in the file `ecb_enc`.

Implementation

```
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>
```

Answers to questions

Is it possible to tell which message has been encrypted based only on the analysis of the ecb-enc, ecb-plain-1, ecb-plain-2 files ? Why, explain ? Record the answer with reasons in the report.

- Based only on analysis of the files `ecb_enc, ecb_plain_1 and ecb_plain_2.` we cannot conclusively determine which plaintext file was used for encryption.
- Justification: No repetition of blocks: In ECB mode, identical blocks in the plaintext mean that ciphertexts will contain the same blocks in the same places. However, in this case, the first block in the ciphertext `(ecb_enc)` is different from the first blocks in both plaintext files `(ecb_plain_1 and ecb_plain_2)`, suggesting that the ciphertext may not originate from either of the two files based on this analysis alone. Lack of block repeatability: For ECB mode, block repetition in plaintext is a key indicator, but there are no repeated blocks in this analysis. This means that the hexadecimal analysis of the first blocks alone is not sufficient to unambiguously indicate which plaintext file has been encrypted.
- Further steps In order to determine more precisely which plaintext file has been encrypted, it would be necessary to continue the analysis at the level of consecutive blocks in the ciphertext and plaintext files. In particular, if the ciphertext contains repeated fragments, this could indicate repeated blocks in one of the plaintext files. However, based on the current analysis, we cannot clearly indicate which file was used.