File encryption

Submission deadline:	**2023-04-25 23:59:59**
Evaluation:	**6.0000**
Max. assessment:	**6.0000** (Without bonus points)
Submissions:	7 / 20 Free retries + 10 Penalized retries (-10 % penalty each retry)
Advices:	1 / 5 Advices for free + 5 Advices with a penalty (-10 % penalty each advice)

Your task is to implement two (or more) functions (not the whole program) that can encrypt and decrypt an image file in **TGA** format.

For our task, we will consider a simplified form of the image:

- Mandatory header: 18 bytes - we do not modify these bytes in any way. We copy them into the encrypted image.
- Optional part of the header: the size is calculated from the mandatory part of the header - we will consider this part of the header as image data, i.e., without changes, we will **encrypt it together** with the image data.
- Image data: rest.

The parameters of the functions implemented by you are:

```
bool encrypt_data (const string & in_filename, const string & out_filename, crypto_config & config)
```

- `in_filename` - input file name,
- `out_filename` - output filename,
- `config` - `crypto_config` data structure is described below.
- The return value is `true` on success, `false` otherwise. Failure occurs if the file is invalid in some way (missing mandatory header, fails to open, read, write, …) or fails to correct the invalid configuration of `crypto_config`.

The `decrypt_data` function uses the same interface but performs the inverse operation to encryption. So the mandatory part of the header, which is **not** encrypted, will be copied, then the rest of the file will be decrypted in the same way as the encryption. But in this case, we expect to pass a valid decryption key and IV (if needed). If we do not have these parameters, we cannot decrypt the data and the program should report an error (`return false`).

The `crypto_config` data structure contains the following:

- selected block cipher entered using its name,
- secret encryption key and its size,
- initialization vector (IV) and its size.

During encryption, the following problem can occur: if the encryption key (or IV) is insufficient (that is, their length is not at least as large as required by the selected block cipher or is completely missing), they must be safely generated. If the entered block cipher does not need an IV (and doesn't have to be given to you), do not generate a new IV! Don't forget to save any generated keys and IVs in the passed config structure!

The following encryption functions will come in handy:

- `EVP_EncryptInit_ex`, or `EVP_DecryptInit_ex`,
- `EVP_EncryptUpdate`, or `EVP_DecryptUpdate`,
- `EVP_EncryptFinal_ex`, or `EVP_EncryptFinal_ex`.

You can see what other features you could (and should) use in the OpenSSL documentation. Hint: Isn’t there a more general function that covers these functions?

By default, block ciphers have padding enabled, and therefore the length of the resulting encrypted file may be longer than the original. This is desired (and expected behavior in tests), and you **shouldn’t** change it.

In the test environment, you will be limited not only by time but also by the size of the available memory; your program may be forcibly terminated. Try not to use the heap unnecessarily, or ideally, don’t use it at all. In 90 % of cases, you will be fine with just the stack.

Submit a source file that contains the implementation of the required `encrypt_data` and `decrypt_data` functions. You can also add your other supporting functions to the source file, which are called from `encrypt_data` and `decrypt_data`. The function will be called from the test environment, so it is important to follow the specified interface of the function exactly.

Use the code from the attached archive below as the basis for the implementation. The sample contains the test function `main`; the specified values are used in the basic test. Note that the header file includes and the `main` function are wrapped in a conditional translation block (`#ifdef/#endif`). Please keep the conditional translation blocks in the source file. Conditional translation will simplify your work. You can run and test the program normally when compiling on your computer. When compiling on Progtest, the `main` function and the header files will “disappear,” so it will not conflict with the test environment’s header files and the `main` function.

Comment:

- **ATTENTION!** Submitted task on Progtest is not a guarantee of a completed task! You can get more information from your teacher.

- Treat file operations with care. The test environment intentionally tests your implementation for non-existent, unreadable files or files with incorrect data content.
- During implementation, you can use C and C++ interfaces for working with files, the choice is yours. STL structures can also be used.
- There are examples in the task specification. The attached archive contains a set of test files and their corresponding equivalents encrypted with some block cipher.
- Input and output files can be large, larger than the size of available memory. Therefore, when working with files, we generally process data continuously. It is unreasonable to load the entire input file into memory and then process it in memory. The last test checks the memory requirements of your solution. It will fail if you try to keep entire files or large parts of them in memory at once.
- Remember that even if some sub-function fails, you must properly handle the dynamically acquired resources.
- Encryption key and IV are generally "some" data, so working with them like ASCII strings is a path to destruction (see exercise).
- Encryption as such is deterministic, but key generation should not be. The test environment checks the correctness of the encryption function by trying to decrypt your encrypted file and comparing the result with expectations (there is a binary match with the original file).
- For easier implementation, the `crypto_config` structure contains smart pointers (`std::unique_ptr`). For their creation, using the `std::make_unique` function is convenient. If you need to pass a raw pointer somewhere, use the `get()` class method.

Sample data: **Download**

Reference

- **Evaluator: computer**
  - Program compiled
  - Test 'Zakladni test se soubory z ukazky': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.015 s (limit: 5.000 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Nespravne vstupy': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.016 s (limit: 4.985 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Nahodny test - sifrovani': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 1.000 s (limit: 4.969 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Nahodny test - desifrovani': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.481 s (limit: 3.969 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Test osetreni I/O chyb': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.115 s (limit: 3.488 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Test pametove narocnosti': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.530 s (limit: 10.000 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Test nahodnymi hodnotami + test prace s pameti - sifrovani': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.876 s (limit: 9.470 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Test nahodnymi hodnotami + test prace s pameti - desifrovani': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.481 s (limit: 8.594 s)
    - Mandatory test success, evaluation: 100.00 %
  - Overall ratio: 100.00 % (= 1.00 * 1.00 * 1.00 * 1.00 * 1.00 * 1.00 * 1.00 * 1.00)
- Retries penalty: 120.00 % (= (32 - 20) * 10 %)
- Total percent: -20.00 % (= 1.00 * -0.20)
- Total points: -0.20 * 6.00 = -1.20

SW metrics:

	Total	Average	Maximum	Function name
Functions:	**8**	--	--	--
Lines of code:	**90**	**11.25 ± 6.83**	**24**	`cipher_data`
Cyclomatic complexity:	**40**	**5.00 ± 4.87**	**13**	`validate_config`

7 **2023-04-25 17:55:20** **Download**

Submission status: Evaluated
Evaluation: 6.0000

- **Evaluator: computer**
  - Program compiled

- Test 'Basic test with example files': success
  - result: 100.00 %, required: 100.00 %
  - Total run time: 0.009 s (limit: 5.000 s)
  - Mandatory test success, evaluation: 100.00 %
- Test 'Invalid input': success
  - result: 100.00 %, required: 100.00 %
  - Total run time: 0.011 s (limit: 4.991 s)
  - Mandatory test success, evaluation: 100.00 %
- Test 'Random test - encryption': success
  - result: 100.00 %, required: 100.00 %
  - Total run time: 1.062 s (limit: 4.980 s)
  - Mandatory test success, evaluation: 100.00 %
- Test 'Random test - decryption': success
  - result: 100.00 %, required: 100.00 %
  - Total run time: 0.515 s (limit: 3.918 s)
  - Mandatory test success, evaluation: 100.00 %
- Test 'Test I/O failures': success
  - result: 100.00 %, required: 100.00 %
  - Total run time: 0.119 s (limit: 3.403 s)
  - Mandatory test success, evaluation: 100.00 %
- Test 'Memory efficiency test': success
  - result: 100.00 %, required: 100.00 %
  - Total run time: 0.568 s (limit: 10.000 s)
  - Mandatory test success, evaluation: 100.00 %
- Test 'Random input test + correct memory usage test - encryption': success
  - result: 100.00 %, required: 100.00 %
  - Total run time: 0.927 s (limit: 9.432 s)
  - Mandatory test success, evaluation: 100.00 %
- Test 'Random input test + correct memory usage test - decryption': success
  - result: 100.00 %, required: 100.00 %
  - Total run time: 0.517 s (limit: 8.505 s)
  - Mandatory test success, evaluation: 100.00 %
- Overall ratio: 100.00 % (= 1.00 * 1.00 * 1.00 * 1.00 * 1.00 * 1.00 * 1.00 * 1.00)

- Advices used: 1
- Penalty due to advices: None (1 <= 5 limit)
- Total percent: 100.00 %
- Total points: 1.00 * 6.00 = 6.00

SW metrics:

	Total	Average	Maximum	Function name
Functions:	**5**	--	--	--
Lines of code:	**173**	**34.60 ± 45.23**	**119**	`execute`
Cyclomatic complexity:	**46**	**9.20 ± 10.44**	**26**	`execute`

6	**2023-04-25 17:51:00**	**Download**
Submission status:	Evaluated	
Evaluation:	6.0000	

- **Evaluator: computer**
  - Program compiled
  - Test 'Basic test with example files': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.016 s (limit: 5.000 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Invalid input': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.018 s (limit: 4.984 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Random test - encryption': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 1.055 s (limit: 4.966 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Random test - decryption': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.518 s (limit: 3.911 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Test I/O failures': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.119 s (limit: 3.393 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Memory efficiency test': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.564 s (limit: 10.000 s)
    - Mandatory test success, evaluation: 100.00 %
  - Test 'Random input test + correct memory usage test - encryption': success
    - result: 100.00 %, required: 100.00 %
    - Total run time: 0.921 s (limit: 9.436 s)

        - Mandatory test success, evaluation: 100.00 %
    - Test 'Random input test + correct memory usage test - decryption': success
        - result: 100.00 %, required: 100.00 %
        - Total run time: 0.513 s (limit: 8.515 s)
        - Mandatory test success, evaluation: 100.00 %
    - Overall ratio: 100.00 % (= 1.00 * 1.00 * 1.00 * 1.00 * 1.00 * 1.00 * 1.00 * 1.00)
- Advices used: 1
- Penalty due to advices: None (1 <= 5 limit)
- Total percent: 100.00 %
- Total points: 1.00 * 6.00 = 6.00

SW metrics:

	Total	Average	Maximum	Function name
Functions:	**5**	--	--	--
Lines of code:	**173**	**34.60 ± 45.23**	**119**	`execute`
Cyclomatic complexity:	**46**	**9.20 ± 10.44**	**26**	`execute`

5	**2023-04-25 17:49:53**	**Download**
Submission status:	Evaluated	
Evaluation:	0.0000	

- **Evaluator: computer**
    - ☐ Compile in 'basic' mode failed.
- Total percent: 0.00 %
- Total points: 0.00 * 6.00 = 0.00

4	**2023-04-25 17:49:12**	**Download**
Submission status:	Evaluated	
Evaluation:	0.0000	

- **Evaluator: computer**
    - Compile in 'basic' mode failed. **[Unlock advice (1377 B)]**
- Total percent: 0.00 %
- Total points: 0.00 * 6.00 = 0.00

3	**2023-04-25 17:47:58**	**Download**
Submission status:	Evaluated	
Evaluation:	0.0000	

- **Evaluator: computer**
    - Compile in 'basic' mode failed. **[Unlock advice (1377 B)]**
- Total percent: 0.00 %
- Total points: 0.00 * 6.00 = 0.00

2	**2023-04-25 17:47:04**	**Download**
Submission status:	Evaluated	
Evaluation:	0.0000	

- **Evaluator: computer**
    - Compile in 'basic' mode failed. **[Unlock advice (1377 B)]**
- Total percent: 0.00 %
- Total points: 0.00 * 6.00 = 0.00

1	**2023-04-25 15:55:43**	**Download**
Submission status:	Evaluated	
Evaluation:	0.0000	

- **Evaluator: computer**
    - Program compiled
    - Test 'Basic test with example files': failed
        - result: 0.00 %, required: 100.00 %
        - Total run time: 0.013 s (limit: 5.000 s)
        - Mandatory test failed, evaluation: 0.00 %
        - Failed (invalid output) **[Unlock advice (67 B)]**
        - Failed (invalid output) **[Unlock advice (99 B)]**
        - Failed (invalid output) **[Unlock advice (21.10 KiB)]**

        - Failed (invalid output) **[Unlock advice (80 B)]**
        - Failed (invalid output) **[Unlock advice (21.11 KiB)]**
        - Failed (invalid output) **[Unlock advice (21.10 KiB)]**
        - Failed (invalid output) **[Unlock advice (80 B)]**
        - Failed (invalid output) **[Unlock advice (96 B)]**
    - Overall ratio: 0.00 %
- Total percent: 0.00 %
- Total points: 0.00 * 6.00 = 0.00

SW metrics:

	Total	Average	Maximum	Function name
Functions:	**5**	--	--	--
Lines of code:	**178**	**35.60 ± 44.57**	**119**	`execute`
Cyclomatic complexity:	**46**	**9.20 ± 10.44**	**26**	`execute`