

Comparison of Symmetric Ciphers in OpenSSL

HW1 - CNS Sapienza

Valerio Coretti 1635747

2019-11-4

1 Introduction

Data encryption procedures are divided into two categories: *Symmetric encryption* and *Asymmetric encryption*. These techniques depending on the type of security keys used to encrypt/decrypt. In this paper we focused in Symmetric Encryption. In this type of encryption the sender and the receiver first agree on a secret "*shared*" key, then they use this secret key to encrypt and decrypt their messages.

Block Ciphers are widely used in symmetric encryption. A block cipher consists of an encryption algorithm and a decryption algorithm. These algorithms take a fixed-length messages, called *blocks*, a variable-length *key*, and through *Operating Modes*, generate a ciphertext. Modes of operations are the techniques used to encrypt messages larger than a single block.

In the following sections, we will present a fair comparison between four of the most common Block Ciphers: *RC2*, *DES*, *Blowfish* and *AES*, based on three different operating modes: *CBC*, *ECB* and *OFB*. In particular we will try to answer to this question: given a pair (cipher, operating mode) is encryption faster or slower than decryption?

2 Preliminary Choices

To answer the above question we have chosen to measure the encryption/decryption speeds on files of different sizes: 1MB, 10MB, 100MB, 300MB. The main parameter on which we based our analysis is the *Average Speed Ratio*.

$$AverageSpeedRatio = (AvgSpeedEncryption)/(AvgSpeedDecryption)$$

The average is calculated on five encryption and five decryption of the same file.

Moreover the following table shows the algorithms settings used in this experiment.

Cipher	Key Size (Bits)	Block Size (Bits)
AES	128	128
DES	64	64
Blowfish	128	64
RC2	128	64

The comparison has been conducted with the OpenSSL library using C language with Linux Mint and over an Intel core i5 64bit processor with 4GB of RAM.

2.1 OpenSSL: Standard Function vs EVP interface

OpenSSL is a software library for applications that secure communications over computer networks against eavesdropping or need to identify the party at the other end. *Wikipedia*

This library has two main sets of function to perform the various cipher algorithms with the related operating modes: *Standard Function* and *EVP interface*. Standard Function are the first set of instructions released by openssl and in fact, as we will see, these are slow and not optimized, especially with cipher AES and CBC mode. Instead the EVP functions provide a high level interface to cryptographic functions.

In this paper first we will discuss the results with the standard functions and then we will see the improvements obtained with EVP.

3 Evaluation Methodology

We have used two Simulations Programs, one for Standard Function and one for EVP. Both take one input parameter that are the file to encrypt and returns all the times-speed for evrey ciphers and related operating modes. Key and IV used is pseudo-randomly generated. Example of output (only for AES):

```

*** READING FILE ***
Length of file = 102096027 Bytes
*** END READING FILE ***

***** Cipher Algorithm: AES *****

+++++++ Operative Mode: CBC
ENCRYPTING Time ==> 0.130107
DECRYPTING Time ==> 0.055643
SPEED RATIO =====> 2.338246

+++++++ Operative Mode: ECB
ENCRYPTING Time ==> 0.031195
DECRYPTING Time ==> 0.033019
SPEED RATIO =====> 0.944759

+++++++ Operative Mode: OFB
ENCRYPTING Time ==> 0.117134
DECRYPTING Time ==> 0.116244
SPEED RATIO =====> 1.007656

```

4 Simulation Results

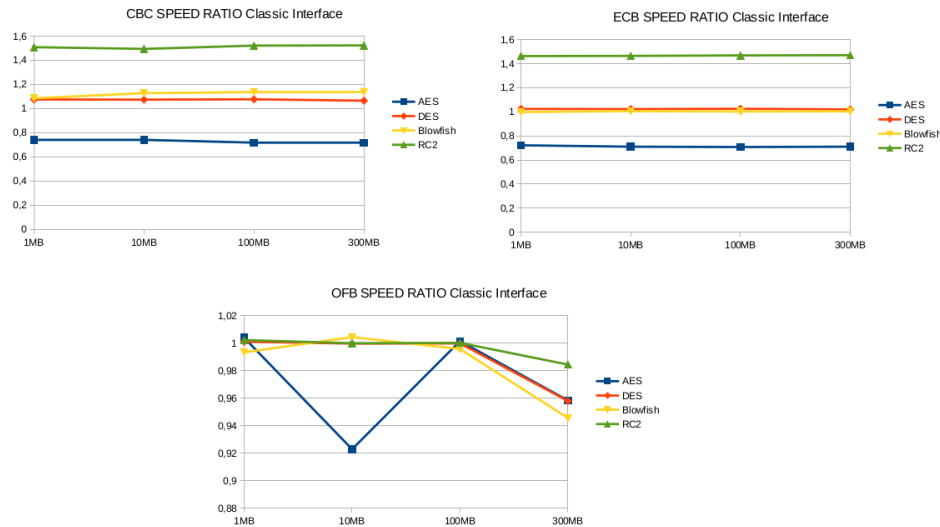
This section will discuss graphics results obtained from running the simulation program for Standard functions and EVP. We will show graphics three different graphics, one for every operating modes.

4.1 Standard functions results

We begin our analysis with CBC. From the graph we immediately notice a surprising thing: AES has the worst speed ratio of all since it is far less than 1. This means that AES with CBC takes longer to decrypt than to encrypt, but as we have also seen during the course this is not possible since CBC provide the parallelization in the decryption process and therefore this should be faster than the encryption.

ECB instead provides parallelization in both encryption and decryption but according to our results AES behaves in the same way as seen with CBC. This is really a strange anomaly. Finally we see that, as expected, with OFB, AES behaves correctly using almost the same time for both decryption and encryption.

For the other ciphers the results are more or less what we expected to see. DES provides the same times for both encryption and decryption as Blowfish, though it is worth remembering that the latter is much faster than DES and slightly slower than AES. RC2 is by far the worst cipher in terms of speed and as we see from the results it uses about half in the decryption compared to encryption.



4.2 EVP results

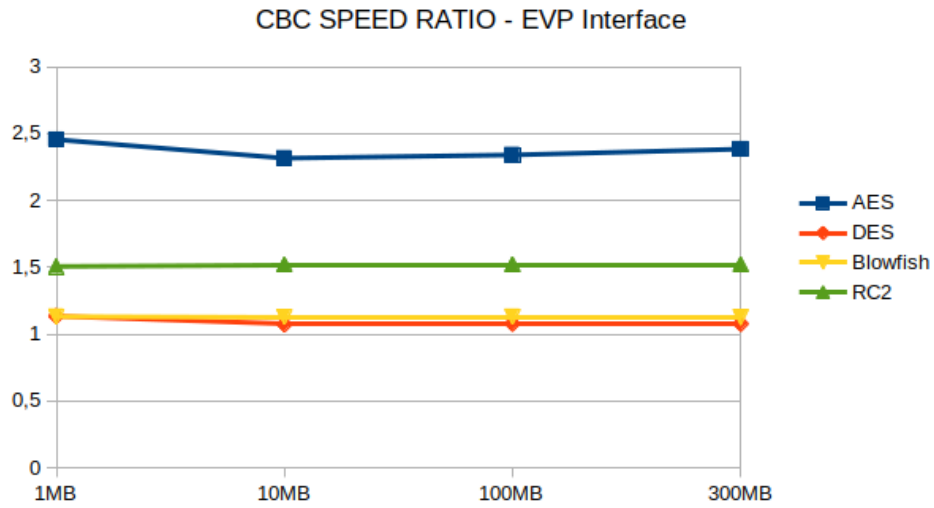
Let's now analyze the results obtained with EVP. The first graph immediately shows us a substantial difference with the previous results, the speed ratio of AES with CBC is close to 2,5. This is a much more accurate and real result than the one seen previously.

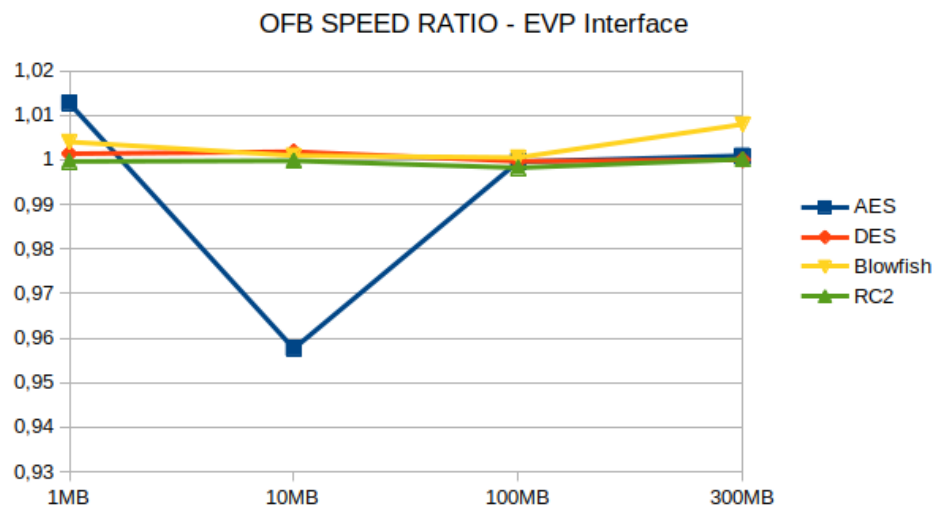
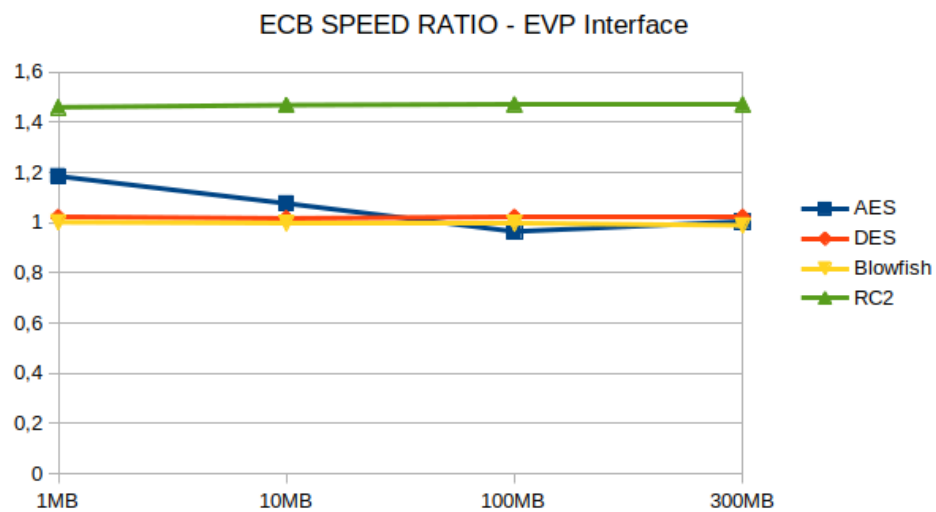
What happens when changing from CBC to ECB is the same result, ie AES with the ECB mode takes the same time both to decrypt and to encrypt. Finally we can see from the last graph that with OFB the speed ratio of the 4 ciphers does not change with respect to before, and this makes us feel a little quieter. In the next section we will explain the reason for these improvements in AES performance using the EVP interface and finally in the conclusion we will try to answer the initial question.

4.2.1 Why EVP is the right way

As we said at the beginning, evp is a high level interface. In fact from the results seen, the performances improve. But this happens above all for AES, why? Because EVP takes advantage of the AES-NI native instructions. To understand how AES-NI works, you need to think that to implement a crypto algorithm, we usually just express a combination of such basic operations additions, multiplications, XORs, and so on, and the microprocessor activates its adders, multipliers, and XOR circuits in the prescribed order.

AES native instructions take this to a whole new level by providing developers with dedicated assembly instructions that compute AES. Instead of coding an AES round as a sequence of assembly instructions, when using AES-NI, you just call the instruction AESENC and the chip will compute the round for you. Native instructions allow you to just tell the processor to run an AES round instead of requiring you to program rounds as a combination of basic operations. In fact running the simulation program we can see also that the speed of AES is higher than that with the standard functions. About the speed ratio evp is optimized and implements the parallelization of operations.





5 Conclusion

Everything we have seen brings us to the following considerations: when you decide to encrypt or decrypt a file using openssl it is always good to use the EVP interface because it increases performance and is optimized for AES. Comparing the different ciphers we have seen that AES is by far the fastest, after which in order we find Blowfish DES and RC2. The latter is the one that has the worst performance. By changing the operating modes the speed ratio changes, in particular CBC makes the decryption faster, especially for AES, while OFB takes the same time both to decrypt and to encrypt whatever is the cipher used. ECB is the fastest of the three. Finally, changing the file size does not change the results significantly.

References

- [1] *Block cipher mode of operation* - Wikipedia.
https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation
Accessed: 2019-11-4.
- [2] Jean - Philippe Aumasson. *SERIOUS CRYPTOGRAPHY A Practical Introduction to Modern Encryption*. Copyright 2018 by Jean-Philippe Aumasson.
- [3] OpenSSL Documentation. <https://www.openssl.org/docs/manmaster/man3/>