Comparison of symmetric ciphers in OpenSSL

HW1 - CNS Sapienza

Giulia Muscarà 1743261

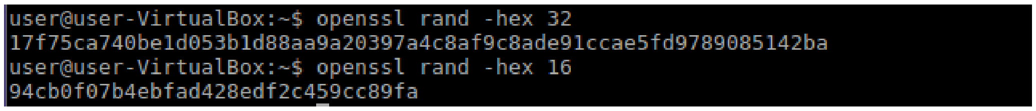
November 4, 2019

1 Objectives

The purpose of this report was to compare different symmetric ciphers, such as AES, DES, RC4 and Blowfish, and the relative operating modes in terms of performance when encrypting and decrypting variable size files, using OpenSSL library. OpenSSL is a cyptography toolkit implementing the Secure Sockets Layer (SSL v2 and v3) and Transport Layer Security network protocols, as well as the related cryptography standards. Among its functions, it allows to perform encryption and decryption with a variety of ciphers.

2 Approach

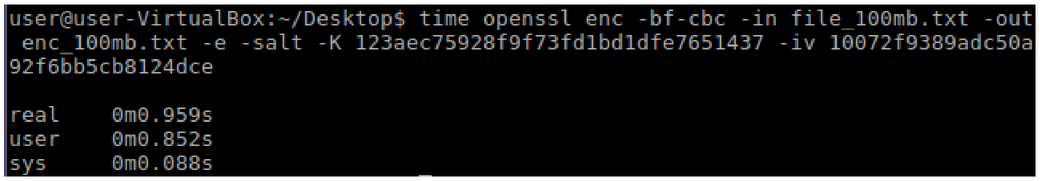
To carry out the analysis, the commands were executed on a scriptable Bash shell running on a Lubuntu virtual box, as it provides a fast and lightweight operating system with a clean and easy-to-use user interface.
The version of OpenSSL that was adopted is the 1.0.2g.
For each tested cipher, three rounds of encryption and decryption for every analyzed operating mode were executed, keeping track of the average running times and comparing the results in charts. Tests were made on four files that were randomly generated setting the desired size: one file of 100 kB, one of 1 MB, one of 10 MB and another of 100 MB, in order to monitor how the encryption and decryption speeds would vary with respect to the variation of the file's size. All the keys, salts and initialization vectors used during the process were randomly generated as well, using the "rand" command, used in Fig.1, and specifying the desired length of the output each time.



```
user@user-VirtualBox:~$ openssl rand -hex 32
17f75ca740be1d053b1d88aa9a20397a4c8af9c8ade91ccae5fd9789085142ba
user@user-VirtualBox:~$ openssl rand -hex 16
94cb0f07b4ebfad428edf2c459cc89fa
```

Figure 1: Pseudo-random bytes generation

The encryption and decryption times were recorded using the bash shell's primitive *time*, that runs the associated command and reports the system resource usage. Most information shown by *time* is derived from the system call *wait3*. As shown in the example below, the command outputs real, user and system elapsed time but the reported measures will be referred to user time, as it is the amount of CPU time spent in user-mode code within the process. The example also shows the command "openssl enc", used to encrypt a file with the option "-e" and to decrypt with the option "-d" using a specific cipher, operating mode, key and initialization vector. The salt was not specified in the example as the command "-salt" automatically uses a randomly generated salt.



```
user@user-VirtualBox:~/Desktop$ time openssl enc -bf-cbc -in file_100mb.txt -out
 enc_100mb.txt -e -salt -K 123aec75928f9f73fd1bd1dfe7651437 -iv 10072f9389adc50a
92f6bb5cb8124dce

real	0m0.959s
user	0m0.852s
sys	0m0.088s
```

Figure 2: Example of time primitive usage

2.1 Symmetric ciphers

In symmetric ciphering, the key that deciphers the ciphertext is the same as the key enciphers the plain text. This key is often referred to as the secret key. The keys that were used were randomly generated and used for both encryption and decryption. The following examples of symmetric ciphers will be used in the experiment.

2.1.1 Triple DES

DES is a symmetric block cipher using 64-bit blocks and a 56-bit key. Initially developed at IBM, it was approved by the NBS (National Bureau of Standards) in 1977 but, because of the short key length, was discarded as too weak for most applications. Triple DES is a more secure version of DES, applying the DES algorithm three times to each data block. This standard offers three key options. The first option is based on an independent choice of the three keys, which offers the strongest level of encryptions. The second keying option is to choose K_1 equal to K_3, but to keep K_2 independent from the others. As a last option, three identical keys can be used.

2.1.2 AES

AES is based on symmetric block cipher of 128-bit block size and uses 128, 192 or 256 bits key length. It operates using a state of 128 bits arranged in a 4 x 4 column major order matrix of bytes and computes 14 rounds for 256-bit key length. Each round changes the state and xors the block with the round key. AES is resistant to all well known attacks, as breaking the full 14 rounds is believed to be not computable in plynomial time. In the example, AES-256 were used as it is the strongest algorithm.

2.1.3 Blowfish

Blowfish is a symmetric key block cipher with key length variable from 32 to 448 bits and block size of 64 bits. Its structure is fiestal network. The variable length of the key makes it ideal for both domestic and commercial use. It was designed by Bruce Schneier and published in 1993 as a fast, free alternative to existing encryption algorithms: it has, indeed, free license and is freely available for all uses. Blowfish is considered to be robust against known attacks.

2.1.4 RC4

The RC4 Encryption Algorithm is a symmetric stream cipher algorithm requiring a secure exchange of a shared key. Developed in 1987 by Ron Rivest, it is chracterized by variable key length and it works in synchronous environments. To generate the key stream, a pseudo-random stream of bits, it exploits a 256-bytes S-box, initialized with a variable length key, through the key-scheduling algorithm. Then the bit stream is generated by a pseudo-random algorithm and it is then combined with the next byte of the plaintext by using bit-wise exclusive-or.
While being a simple and fast algorithm, RC4 is considered to be weak. Indeed, the generated sequence is not truly random and will eventually repeat, and it was proven it is possible to obtain information about the key stream and therefore about the plaintext because of the weak key schedule.

3 Results

In the tables below the results of the experiment were recorded, for each file size and for each pair of cipher and operating mode. The average ecryption and decryption times were calculated on three trials for every measure, executing the script thrice.
The speed ratio column refers to the ratio of the average speed of encryption and the average speed of decryption. In the case that speed ratio is greater than or equal to 1, the encryption is faster than descryption or takes the same amount of time for that given pair of cipher and operating mode, and viceversa.

Table 1: Running times for a 100 kB file

Cipher	Average encryption time (ms)	Average decryption time (ms)	Speed ratio
des-ede3-cbc	8,00	6,67	1,19
des-ede3	6,67	8,00	0,83
aes-256-cbc	4,00	4,00	1,00
aes-256-ecb	4,00	4,00	1,00
rc4	4,00	4,00	1,00
rc4-40	4,00	4,00	1,00
bf-cbc	4,00	4,00	1,00
bf-ecb	4,00	4,00	1,00

Table 2: Running times for a 1 MB file

Cipher	Average encryption time (ms)	Average decryption time (ms)	Speed ratio
des-ede3-cbc	49,33	45,33	1,09
des-ede3	45,33	49,33	0,92
aes-256-cbc	6,67	4,00	1,67
aes-256-ecb	4,00	4,00	1,00
rc4	5,33	6,67	0,79
rc4-40	5,33	6,67	0,79
bf-cbc	12,00	10,67	1,12
bf-ecb	14,67	10,67	1,37

Table 3: Running times for a 10 MB file

Cipher	Average encryption time (ms)	Average decryption time (ms)	Speed ratio
des-ede3-cbc	394,67	374,67	1,05
des-ede3	393,33	389,33	1,01
aes-256-cbc	16,00	9,33	1,71
aes-256-ecb	9,33	10,67	0,87
rc4	20,00	25,33	0,79
rc4-40	20,00	26,67	0,75
bf-cbc	81,33	82,67	0,98
bf-ecb	88,00	90,67	0,97

Table 4: Running times for a 100 MB file

Cipher	Average encryption time (ms)	Average decryption time (ms)	Speed ratio
des-ede3-cbc	3836,00	3705,33	1,04
des-ede3	4062,67	3933,33	1,03
aes-256-cbc	141,33	53,33	2,65
aes-256-ecb	38,67	42,67	0,91
rc4	178,67	161,33	1,11
rc4-40	166,67	177,33	0,94
bf-cbc	854,67	746,67	1,15
bf-ecb	852,00	857,33	0,99

4 Discussion

Encryption and decryption times, given pairs of cipher and operating mode, can be compared basing on the speed ratio reported in the tables. While for the 100 kB file no significant difference between the speeds was observed, it emerges that for bigger files aes-256 in cbc mode performed encryption slower than decryption almost doubling its speed. Another significant result is that rc4 cipher, regardless of the operating mode, proved to be slower during decryption.
Triple des tends to be slow in software implementations as it was designed for hardware implementations in the early '70s and as it executes encryption and decryption three times each time. Indeed, values for 3des both in cbc and ecb mode were high and it recorded the slowest times while the speed ratio can be approximated to 1, showing how there is little to no difference between encryption and decyption times.
Aes-256 performed instead very quickly in both modes, recording both the fastest decryption and encryption times, justifyable by the fact that subBytes and multiplication in MixColumns can be replacede by a lookup table of 256 bytes. Using aes-256 speed ratio change significantly as in cbc mode, it is greater than 1 and in ecb mode is less than 1.
Rc4 is one of the fastest algorithms, with encryption being faster than decryption and, consequently, speed ratio being less than 1.
Blowfish was the second slowest algorithm and encryption was slightly faster than decryption, causing speed ratio to be on average a little greater than 1. As far as aes-256 is concerned, for both encryption and decryption, moving from cbc to ecb mode caused a reduction of processing times. On the other hand, using blowfish in ecb mode took more time than with cbc mode. 3-Des with cbc registered slightly higher time for encryption but higher times for decryption in ecb. Cbc mode generally tends to affect the speed ratio making it grow greater than 1. As shown in the graphs in Fig.3 and Fig.4, as the file size increases, also encryption and decryption times do, keeping speed radio approximately constant. In both cases aes-256 in ecb mode was the fastest algorithm and triple des in ecb mode recorded the slowest times.

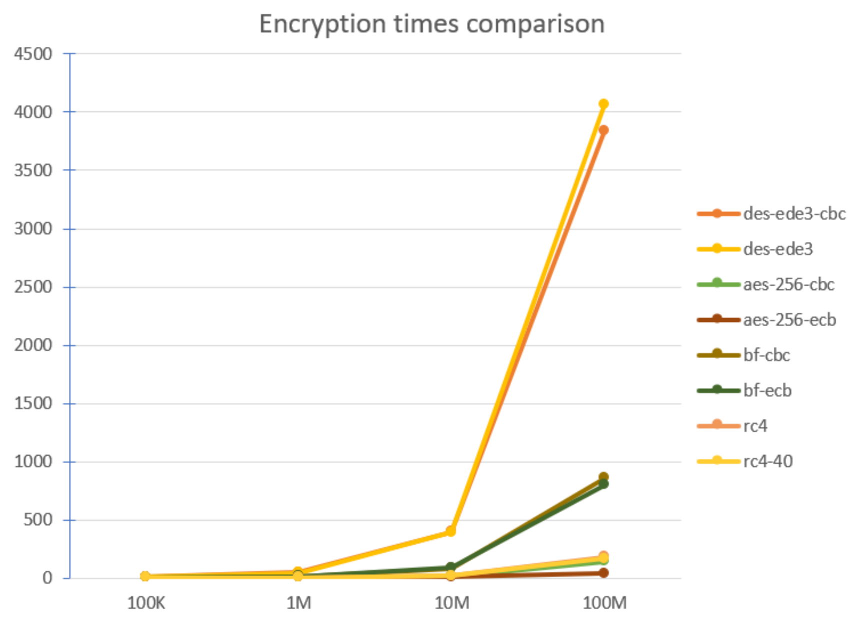


Figure 3: Encryption times

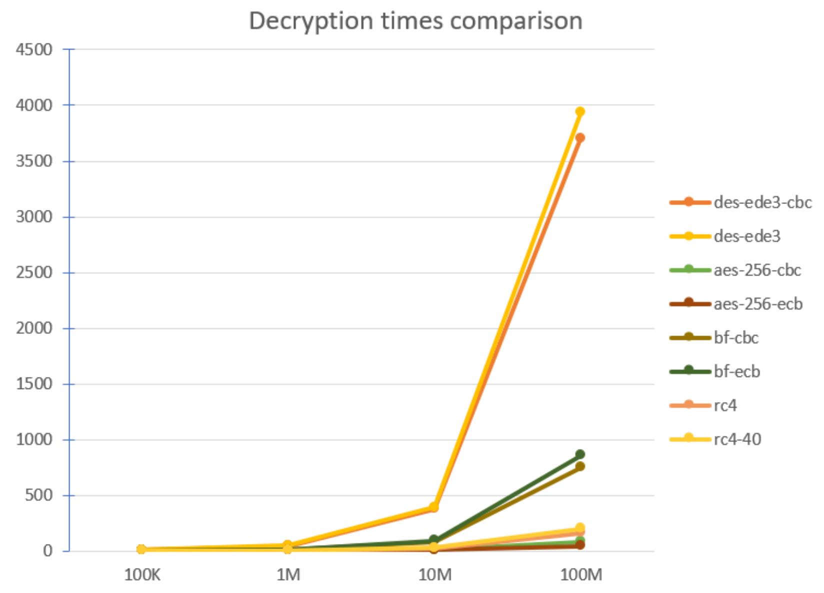


Figure 4: Decryption times

5 Conclusions

All the above analyzed cryptographic algorithms have their strengths and weaknesses. Depending on the demands of the application that will be used, the most suitable cryptographic algorithm should be adopted, according to its features. As it emerges from the withdrawn data, aes-256 algorithm would be a better choice in case the application requires fast encryption and decryption, since it recorded the shortest time among all the tested ciphers.