My crypto libraries

HW4-CNS Sapienza

Manuel Ivagnes 1698903

28/11/19

1 Introduction

Until now we have only used OpenSSL to perform the encryption and decryption operations. The purpose of this homework is to explore some valid free-license alternatives to compare.

For the tests I have chosen the Python and C languages, the first one provides an high level and user friendly interface, the second one is the best to work at low level and (in general) provides the best performances.

To notice that the descriptions below mostly consider the symmetric cryptographic functions provided by the libraries.

2 Libraries

2.1 Cryptography

Cryptography is a package which provides cryptographic recipes and primitives for **Python**. It includes both high level recipes and low level interfaces to common cryptographic algorithms such as symmetric ciphers, message digests, and key derivation functions.

The first one is safe and easy to use and don't require developers to make many decisions. Indeed, *"Fernet"* provides an easy way to implement the authenticated symmetric encryption just by taking the key (It uses the AES-256 cipher with CBC mode and HMAC). An example of implementation is provided in the code.

This first method is very user friendly, but, it is pretty slow and does not provide configuration options. Using the "hazardous materials layer", instead, it is possible to access to all the funtionalities.

The documentation is provided at the following link:
https://cryptography.io/en/latest/

The source code is provided at the following link:
https://github.com/pyca/cryptography

2.2 PyCryptodome

PyCryptodome is a self-contained **Python** package of low-level cryptographic primitives. It is a fork of PyCrypto which is not updated anymore, making it useful for compatibility with old programs. It has a good documentation and it is easy to use but the symmetric encryption speed is poor compared to other alternatives.

The documentation is provided at the following link:
https://pycryptodome.readthedocs.io/en/latest/

The source code is provided at the following link:
https://github.com/Legrandin/pycryptodome

2.3 OpenSSL

OpenSSL contains an open-source implementation of the SSL and TLS protocols. The core library, written in the **C** programming language, implements basic cryptographic functions and provides various utility functions. It is widely used by Internet servers, including the majority of HTTPS websites. It is very fast and used as base for many other libraries.

In my opinion the basic documentation is not as good as the other libraries, but it is possible to find many other sources on the web, for instance the OpenSSL wiki:
https://wiki.openssl.org/index.php/Main_Page

The documentation is provided at the following link:
https://www.openssl.org/docs/

The source code is provided at the following link:
https://github.com/openssl/openssl

2.4 WolfCrypt

The wolfCrypt cryptography engine is a lightweight crypto library written in ANSI **C** and targeted for embedded, RTOS, and resource-constrained environments - primarily because of its small size, speed, and feature set.

It is supported by many different operating systems and the documentation is really well done. Unfortunately it is meant for small inputs, while the size of the input increase there is a huge decrease of performances.

The documentation is provided at the following link:
https://www.wolfssl.com/docs/

The source code is provided at the following link:
https://github.com/wolfssl/wolfssl

3 Measures

3.1 Cryptography

1. Results test txt size 100 KB
    - Encryption time average: 0.000375s
    - Decryption time average: 0.000211s
    - Speed-ratio: 1.7694
2. Results test txt size 500 KB
    - Encryption time average: 0.002014s
    - Decryption time average: 0.000922s
    - Speed-ratio: 2.1831
3. Results test txt size 5 MB
    - Encryption time average: 0.015986s
    - Decryption time average: 0.008228s
    - Speed-ratio: 1.9428

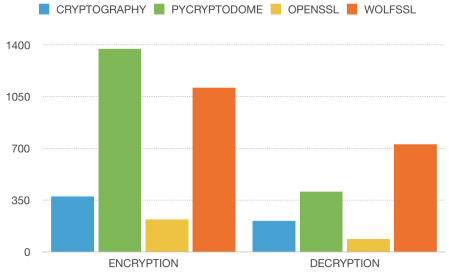
3.2 PyCryptodome

1. Results test txt size 100 KB
    - Encryption time average: 0.001374s
    - Decryption time average: 0.000408s
    - Speed-ratio: 3.3662
2. Results test txt size 500 KB
    - Encryption time average: 0.002885s
    - Decryption time average: 0.002656s
    - Speed-ratio: 1.0858
3. Results test txt size 5 MB
    - Encryption time average: 0.021348s
    - Decryption time average: 0.022346s
    - Speed-ratio: 0.9553

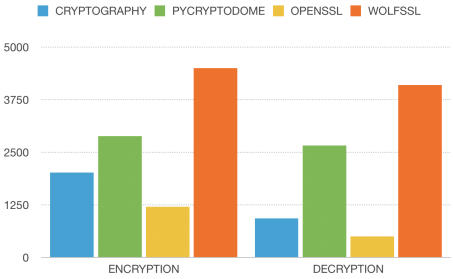
3.3 OpenSSL

1. Results test txt size 100 KB
    - Encryption time average: 0.000221s
    - Decryption time average: 0.000088s
    - Speed-ratio: 2.5113
2. Results test txt size 500 KB
    - Encryption time average: 0.001207s
    - Decryption time average: 0.000500s
    - Speed-ratio: 2.4140
3. Results test txt size 5 MB
    - Encryption time average: 0.011436s
    - Decryption time average: 0.004454s
    - Speed-ratio: 2.5675

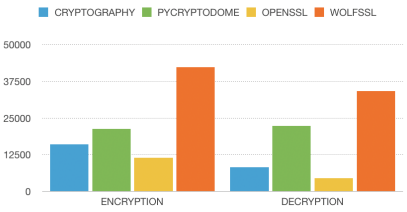
100 KB	ENCRYPTION	DECRYPTION
CRYPTOGRAPHY	375	211
PYCRYPTODOME	1374	408
OPENSSL	221	88
WOLFSSL	1110	728



500 KB	ENCRYPTION	DECRYPTION
CRYPTOGRAPHY	2014	922
PYCRYPTODOME	2885	2656
OPENSSL	1207	500
WOLFSSL	4500	4100



5 MB	ENCRYPTION	DECRYPTION
CRYPTOGRAPHY	15986	8228
PYCRYPTODOME	21348	22346
OPENSSL	11436	4454
WOLFSSL	42269	34147



3.4 WolfCrypt

1. Results test txt size 100 KB
   - Encryption time average: 0.001110s
   - Decryption time average: 0.000728s
   - Speed-ratio: 1.5247
2. Results test txt size 500 KB
   - Encryption time average: 0.004500s
   - Decryption time average: 0.004100s
   - Speed-ratio: 1.0975
3. Results test txt size 5 MB
   - Encryption time average: 0.042269s
   - Decryption time average: 0.034147s
   - Speed-ratio: 1.2378

4 Conclusions

For the test measures I have used the CBC mode mostly to see the differences with parallelization, which seems to disappear using PyCryptodome with relatively big files. Something similar happens with WolfSSL, which in general have bad performances for all the tests, due to the size of the files. Indeed, it is meant to work with small files. The OpenSSL library confirms to be the fastest one, but also Cryptography is surprisingly fast considering it easiness.

References

[1] Course slides

[2] https://wiki.openssl.org/index.php/EVP_Symmetric_Encryption_and_Decryption

[3] Docs linked under the corresponding library