My crypto libraries

HW4 - CNS Sapienza

Matteo Salvino 1708108

28 November 2019

1 Goals

In this fourth CNS homework we are interested in to use some cryptography libraries in order to have useful references for the future. Our test environment will be characterized by two programming languages (Python and C). For each of them we will take two open source cryptography libraries and try to encrypt a binary file with one symmetric algorithm implemented in the first library and to decrypt the obtained ciphertext with the same symmetric algorithm but implemented in the second library. This task is not trivial, because most likely different libraries will have implemented the same symmetric algorithm in different ways. So, our goal is to perform this task, eventually fixing some problems take comes out.

2 Python Environment

For Python programming language we chosen the following cryptography libraries :

cryptography.io It includes low level interfaces to the most common cryptographic algorithms, like symmetric ciphers, message digests and key derivation function. It's currently maintained, indeed the last update dates back to one month ago (version 2.8 released). It is divided in two main levels :

- **Recipes level** : This level is so safe and easy to use because it doesn't require to developers to make a lot of decisions. It's like a black box where a developer just put his input parameters and got the output returned from it.

- **Hazardous level** : His name derived to the fact that developer in this level have more freedom respect to the previous one. In fact, it implements low-level cryptographic primitives, that requires to make decisions and a lot of field knowledge in order to avoid dangerous consequences.

PyCryptodome It is a self-contained Python package of low-level cryptographic primitives. It's a fork of PyCrypto library but implements a lot of improvements respect to it. The most of algorithms provided are implemented in pure Python, whereas only critical features are implemented in C language. It's currently maintained, indeed the last update dates back to few days ago (version 3.9.4 released).

2.1 cryptography > PyCryptodome

To start our experiment lets take a sample jpeg image (size around 350KB) and try to encrypt it via a symmetric cipher with a specific mode of operation implemented in the first library and to decrypt the resulting ciphertext with the same cipher and mode of operation but implemented in the second library. As cipher we chosen *Blowfish* and as mode of operation *CBC*. In order to provide to the cipher a binary plaintext we need to read this image in binary mode. Subsequently, providing an 8 byte (because it should have the same size of block size used by the cipher) initialization vector to CBC and a key, we obtain the corresponding ciphertext. Now, we will provide the same key and iv to the implementation of Blowfish and CBC of second library. Finally, since the implementation are really similar we managed to decrypt the previous ciphertext, getting back our image.

3 C Environment

For C programming language we chosen the following two cryptography libraries :

OpenSSL It implements basic cryptographic functions and provides various useful functions. Last update was released two months ago (version 1.1.1d released). It's a fork of *SSLeay* (an open source SSL implementation), which the authors ended the development when OpenSSL start the development in 1998. His development is mainly paid by people's donations. It's currently released with *Apache 2.0* license.

Libgcrypt It's a cryptographic library based on GnuPG code, that provides a lot of function such as symmetric cipher algorithms, several modes of operation, hash algorithms, etc. It's presently maintained, indeed the last update was released three months ago (1.8.5 version released). It's released under *GNU Lesser General Public License version 2.1+*.

3.1 OpenSSL > Libgcrypt

Now, we want to repeat the previous experiment in C programming language, in order to see if we can encrypt with OpenSSL library and decrypt with Libgcrypt library. Thus, lets consider *AES* with 128 key length as symmetric cipher and *CBC* as mode of operation. Currently, if we encrypt a binary file (our sample image) with AES implemented in the first library and decrypt the corresponding ciphertext with the same symmetric cipher implemented in the other library, we notice that the resulting plaintext is different from the original plaintext. This is due from different cipher implementation of the considered libraries. In fact, after some encryption/decryption test with the first library, we figure out that the corresponding ciphertext is three times bigger than the original plaintext, whereas using the second library the ciphertext is as big as the plaintext. A further difference is that OpenSSL has a function that automatically manage the padding to add to the last plaintext block, if eventually the plaintext size isn't a multiple of cipher's block size. While Libgcrypt doesn't have a function like this, but we will have to manage ourselves the need to add some padding bits. We also tried to build a common interface between these libraries, but we didn't have success because the result depends on the intrinsic mechanisms behind the cipher implementation of each library.

4 Conclusions

This has been a nice homework because lays the foundations to understand and build something bigger. In the future working life probably we will have to manage different libraries that doesn't have a common interface and our goal will be to build this interface in order that all things properly works. Another interesting task would be to encrypt some data using a specific programming language and decrypt it with another programming language. Naturally, this task is harder than the previous one because we need to build a common interface between two different programming languages and not between two different libraries written with the same programming language.