nucosCR Documentation

Release 0.2.1

Oliver Braun, Johannes Eckstein

Dec 21, 2020

Contents

1 Install **3**
1.1 Python Compatibility . . . 3
1.2 Platforms . . . 3

2 Basic Usage **5**
2.1 RSA-Suite . . . 5
2.2 AES-Suite . . . 6
2.3 Copy-Script . . . 6

3 Links **7**

Index **9**

nucosCR is a cryptonize Toolbox based on *pycrypto* and the examples therein. The project aim is to improve the usability of the low level API. It implements up to now only our main needs: RSA and AES.

CHAPTER 1

Install

If you want it simple, go to a console an type

```
pip install nucosCR
```

Otherwise it may also be recommended to install it into a virtual environment. To that end type in a console

```
virtualenv venv
source ./venv/bin/activate
pip install nucosCR
```

1.1 Python Compatibility

The module is compatible with Python 2.7 and Python 3.4/3.5/3.6 It should also work in any other Python Version but not yet tested.

1.2 Platforms

The module is platform independent. Up to now it is tested on linux (ubuntu). Other platforms will follow.

CHAPTER 2

Basic Usage

Short

- There are two classes: *CryptoRSABase* and *CryptoAESBase*
- The principle usage is shown in the tests

2.1 RSA-Suite

To work with the RSA-Suite you should import the following

```
from nucosCR import CryptoRSABase
```

A usual working example for creating a public-private key file would be

```
c = CryptoRSABase()
name = "test_admin"
key1 = c.create_rsa_key(name)
```

Internally the PEM-files are stored in a folder relative to the working directory called *./PEM*. The full key can be read with the function *get_key_by_file*. The *name* is always the reference to the key. It can be any *string*.

After creating the key, it can be checked by

```
key2 = c.get_key_by_file(name)
assertEqual(key1, key2)
```

An example for en- and decoding with the previously generated key would be

```
name = "test_admin"
hexkey = c.get_hex_key(name)
```

(continues on next page)

(continued from previous page)

```
txt = b"my own secret message"
#encryption with
en = c.encrypt(txt, hexkey)
#decryption with
de = c.decrypt(name, en)
assertEqual(txt, de)
```

Note here, that for *encryption* only the public part of the key must be used, which is here represented in hex. For *decryption* the full key is needed, so it is referenced here with its *name*.

2.2 AES-Suite

To work with the AES-Suite you should import the following

```
from nucosCR import CryptoAESBase
```

A usual working example would be

```
c = CryptoAESBase("secret")
text = b"my message"
#encryption with
en = c.encryption(text)
#decryption with
de = c.decryption(en)
self.assertEqual(text, de)
```

The class works internally with *bytes* and so does the decryption produce bytes as a result. For convenience the passed argument in the encryption function may also be *string*.

The class is instanciated with a password (in our example case *secret*). Internally the password is digested with SHA256 into a much longer passphrase.

2.3 Copy-Script

This package provide a copy-script for copying a file or folder to a destination file or folder. During copying the script encrypts the data with the aes-algorithm. If the source is a folder, it is copied recursively into the the destination. The usage is

```
aes-cp [-e ,-d ,-c, -o] source destination
```

-e source destination encryption copy

-d source destination decryption copy

-c file1 file2 check crc

-o overwrite flag (default is not overwrite), if set it overwrites the files in the destination

Before the copy process starts the user is prompted for a password.

CHAPTER 3

Links

- genindex
- search

Index

B

Basic Usage, 5

I

Install, 3