py-crypto-params Documentation

Release 1.0.0

Gian Luca Dalla Torre

December 30, 2015

Contents

Utility function to encrypt - decrypt string using AES symmetric algorithm that is compatible with crypto-js.

CHAPTER 1

Purpose

Harvesting data on the web has become an easy task.

Often, to obtain data stored into a database, a simple script loops on a numeric query parameter (called usually *id*) embedded into an *URL* and it donwloads a lot of useful data.

Another weakness on sites are *Javascript config files* that holds *JSON* with valuable data.

Last but not least, *AJAX call* contains a lot of information and, if unprotected, they can easily looped to obtain all their contents.

How to prevent these flaws? Maybe if the query string or the data is encrypted a lot of those scripts will not work...

CHAPTER 2

How it works

The *CryptoParams* class provide methods to encrypt and decrypt strings using AES algorithm [1]. This way query parameters (but also *JSON responses*) can be obfuscated and read only by the possessors of the encryption key.

This particular implementation, inspired by marcoslin gist is compatible with crypto-js [2] ; this mean that a parameter encoded by a *HTTP server* could be read by *Javascript*. The only caveat is to share (or at least to obfuscate) the key (and the initialization vector) in a safely manner.

If the parameter is only on query string, only the server can translate them (since the key is not exposed), avoiding obnoxious looping scripts that harvest the data.

2.1 Documentation

2.1.1 Installation

This storage is hosted on PyPI. It can be easily installed through *pip*:

```
pip install py-crypto-params
```

2.1.2 Usage

To initialize the encryption - decryption system, the *CryptoParams* class is used:

```
import cryptoparams

cp = cryptoparams.CryptoParams()
```

The initialization without parameters auto generate a 32 bytes key and a 32 bytes initialization vector (as per AES specification).

The generated values are available through these properties:

- *key*
- *iv*

[1] AES is a symmetric encryption - decryption algorithm based on a 32 bytes shared key (and a shared *Initialization Vector*) that can obfuscate parameters and data.

[2] Starting from this GIST, sooner I will implement the *Javascript version of this algorithm* to allow the reading of data sent from the server directly in HTML pages.

CryptoParams class accept custom *key* and *initialization vector* though the properties above and using the constructor:

```
import cryptoparams

cp = cryptoparams.CryptoParams("d0540d01397444a5f368185bfcb5b66b", "a1e1eb2a20241234a1e1eb2a20241234"
```

The requisites to use custom *key* and *initialization vector* are:

- **key** must be a 32 bytes string written in hexadecimal base (it is not meant to be human readable)
- **initialization vector** must be a 32 bytes string written in hexadecimal base (it is not meant to be human readable)

If those requirements are not met a `ValueError` exception will be raised.

Once the class has been initialized, a string could be encrypted using *encrypt(value)* method:

```
encrypted_data = cp.encrypt("aieiebrazorf")
# encrypted_data contains "iW8qzzEWpWRN0NPNoOwu3A=="
```

This function returns a **Base64 encoded string** ready to be used into query strings.

To decrypt a **Base64 encoded string** with data the method used is *decrypt(value)*:

```
decrypted_data = cp.encrypt("iW8qzzEWpWRN0NPNoOwu3A==")
# decrypted_data contains "aieiebrazorf"
```

It is possibile to encrypt and decrypt complex data transofming them into string such as *JSON*:

```
import cryptoparams
import json

original_data = {
    "id": 1065412,
    "user_id": 657
}

data_to_encrypt = json.dumps(original_data)
encrypted_string = cp.encrypt(data_to_encrypt)
decrypted_string = cp.decrypt(encrypted_string)
decrypted_data = json.loads(decrypted_string)
# decrypted_data contains a dict equal to original_data
```

2.1.3 Source and License

Source can be found on GitHub with its included license.

2.2 Class reference

This is the complete class reference for this project

2.2.1 Class *CryptoParams*

This is the class that allows the encryption and decryption of string using AES algorithm:

class cryptoparams.**CryptoParams**(*key=None*, *iv=None*)

Provide encryption and decryption function for strings that use AES symmetric algorithm with CBC (which is compatible with crypto-js).

Padding implemented as per RFC 2315 PKCS#7 page 21

First base implementation for this class is taken from marcoslin gist.

Parameters

- **key** (*str*) – 32 bytes hexadecimal key used to initialize AES algorithm
- **iv** (*str*) – 32 bytes hexadecimal initialization vector used to initialize AES algorithm

Raises ValueError if parameters are incorrect

decrypt(*value*)

Decrypt a Base64 string using AES algorithm (CFB mode)

Parameters **value** (*str*) – Base64 String to decrypt

Returns String that represent the decrypted data with AES algorithm

Return type str

encrypt(*value*)

Encrypt a string using AES algorithm (CFB mode) and encode the result in Base64 to handle it easily

Parameters **value** (*str*) – String to encrypt

Returns Base64 String that represent the binary data encrypted with AES algorithm

Return type str

iv

Initialization vector used by this class

Returns 32 bytes hexadecimal string containing the initialization vector used by the class

Return type str

Raises ValueError if invalid key is provided

key

AES Key used by the class

Returns 32 bytes hexadecimal string representing the key used by the class

Return type str

Raises ValueError if invalid key is provided

C

D

E

I

K