php-crypto-params Documentation

Release 1.0.0

Gian Luca Dalla Torre

January 17, 2016

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\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 Packagist. It can be easily installed configuring Composer:

```
{
    "require": {
        "torre76/php-crypto-params": "1.0.*"
    }
}
```

2.1.2 Usage

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

```
<?php
require __DIR__ . '/vendor/autoload.php';

$cp = new \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:

[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.

- `key`
- `iv`

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

```
<?php
require __DIR__ . '/vendor/autoload.php';

$cp = new \CryptoParams\CryptoParams("d0540d01397444a5f368185bfcb5b66b", "a1e1eb2a20241234a1e1eb2a202
```

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 `‘\\CryptoParams\\CryptoParamsException’` exception will be raised.

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

```
<?php
require __DIR__ . '/vendor/autoload.php';

$cp = new \CryptoParams\CryptoParams("d0540d01397444a5f368185bfcb5b66b", "a1e1eb2a20241234a1e1eb2a202
$encrypted = $cp->encrypt("aieiebrazorf");

// $encrypted 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)`:

```
<?php
require __DIR__ . '/vendor/autoload.php';

$cp = new \CryptoParams\CryptoParams("d0540d01397444a5f368185bfcb5b66b", "a1e1eb2a20241234a1e1eb2a202
$decrypted = $cp->decrypt("iW8qzzEWpWRN0NPNoOwu3A==");

// $decrypted contains "aieiebrazorf"
```

It is possibile to encrypt and decrypt complex data transofming them into string such as *JSON*. Everything that can be serialized to a string can be encrypted and decrypted:

```
<?php
require __DIR__ . '/vendor/autoload.php';

$cp = new \CryptoParams\CryptoParams("d0540d01397444a5f368185bfcb5b66b", "a1e1eb2a20241234a1e1eb2a202
$data = array();
$data["id"] = 1;
$data["description"] = "Description";

$buffer = json_encode($data);
$encrypted = $cp->encrypt($buffer);

$buffer = $cp->decrypt($encrypted);
$data = json_decode($buffer, FALSE);

// $data->id contains 1
// $data->description contains "Description"
```

2.1.3 Source and License

Source can be found on GitHub with its included license.