* [Welcome to CodeIgniter](http://docs.google.com/general/welcome.html)
* [Installation Instructions](http://docs.google.com/installation/index.html)
  + [Downloading CodeIgniter](http://docs.google.com/installation/downloads.html)
  + [Installation Instructions](http://docs.google.com/installation/index.html)
  + [Upgrading From a Previous Version](http://docs.google.com/installation/upgrading.html)
  + [Troubleshooting](http://docs.google.com/installation/troubleshooting.html)
* [CodeIgniter Overview](http://docs.google.com/overview/index.html)
  + [Getting Started](http://docs.google.com/overview/getting_started.html)
  + [CodeIgniter at a Glance](http://docs.google.com/overview/at_a_glance.html)
  + [Supported Features](http://docs.google.com/overview/features.html)
  + [Application Flow Chart](http://docs.google.com/overview/appflow.html)
  + [Model-View-Controller](http://docs.google.com/overview/mvc.html)
  + [Architectural Goals](http://docs.google.com/overview/goals.html)
* [Tutorial](http://docs.google.com/tutorial/index.html)
  + [Static pages](http://docs.google.com/tutorial/static_pages.html)
  + [News section](http://docs.google.com/tutorial/news_section.html)
  + [Create news items](http://docs.google.com/tutorial/create_news_items.html)
  + [Conclusion](http://docs.google.com/tutorial/conclusion.html)
* [Contributing to CodeIgniter](http://docs.google.com/contributing/index.html)
  + [Writing CodeIgniter Documentation](http://docs.google.com/documentation/index.html)
  + [Developer’s Certificate of Origin 1.1](http://docs.google.com/DCO.html)
* [General Topics](http://docs.google.com/general/index.html)
  + [CodeIgniter URLs](http://docs.google.com/general/urls.html)
  + [Controllers](http://docs.google.com/general/controllers.html)
  + [Reserved Names](http://docs.google.com/general/reserved_names.html)
  + [Views](http://docs.google.com/general/views.html)
  + [Models](http://docs.google.com/general/models.html)
  + [Helpers](http://docs.google.com/general/helpers.html)
  + [Using CodeIgniter Libraries](http://docs.google.com/general/libraries.html)
  + [Creating Libraries](http://docs.google.com/general/creating_libraries.html)
  + [Using CodeIgniter Drivers](http://docs.google.com/general/drivers.html)
  + [Creating Drivers](http://docs.google.com/general/creating_drivers.html)
  + [Creating Core System Classes](http://docs.google.com/general/core_classes.html)
  + [Creating Ancillary Classes](http://docs.google.com/general/ancillary_classes.html)
  + [Hooks - Extending the Framework Core](http://docs.google.com/general/hooks.html)
  + [Auto-loading Resources](http://docs.google.com/general/autoloader.html)
  + [Common Functions](http://docs.google.com/general/common_functions.html)
  + [Compatibility Functions](http://docs.google.com/general/compatibility_functions.html)
  + [URI Routing](http://docs.google.com/general/routing.html)
  + [Error Handling](http://docs.google.com/general/errors.html)
  + [Caching](http://docs.google.com/general/caching.html)
  + [Profiling Your Application](http://docs.google.com/general/profiling.html)
  + [Running via the CLI](http://docs.google.com/general/cli.html)
  + [Managing your Applications](http://docs.google.com/general/managing_apps.html)
  + [Handling Multiple Environments](http://docs.google.com/general/environments.html)
  + [Alternate PHP Syntax for View Files](http://docs.google.com/general/alternative_php.html)
  + [Security](http://docs.google.com/general/security.html)
  + [PHP Style Guide](http://docs.google.com/general/styleguide.html)
* [Libraries](http://docs.google.com/index.html)
  + [Benchmarking Class](http://docs.google.com/benchmark.html)
  + [Caching Driver](http://docs.google.com/caching.html)
  + [Calendaring Class](http://docs.google.com/calendar.html)
  + [Shopping Cart Class](http://docs.google.com/cart.html)
  + [Config Class](http://docs.google.com/config.html)
  + [Email Class](http://docs.google.com/email.html)
  + [Encrypt Class](http://docs.google.com/encrypt.html)
  + [Encryption Library](#gjdgxs)
  + [File Uploading Class](http://docs.google.com/file_uploading.html)
  + [Form Validation](http://docs.google.com/form_validation.html)
  + [FTP Class](http://docs.google.com/ftp.html)
  + [Image Manipulation Class](http://docs.google.com/image_lib.html)
  + [Input Class](http://docs.google.com/input.html)
  + [Javascript Class](http://docs.google.com/javascript.html)
  + [Language Class](http://docs.google.com/language.html)
  + [Loader Class](http://docs.google.com/loader.html)
  + [Migrations Class](http://docs.google.com/migration.html)
  + [Output Class](http://docs.google.com/output.html)
  + [Pagination Class](http://docs.google.com/pagination.html)
  + [Template Parser Class](http://docs.google.com/parser.html)
  + [Security Class](http://docs.google.com/security.html)
  + [Session Library](http://docs.google.com/sessions.html)
  + [HTML Table Class](http://docs.google.com/table.html)
  + [Trackback Class](http://docs.google.com/trackback.html)
  + [Typography Class](http://docs.google.com/typography.html)
  + [Unit Testing Class](http://docs.google.com/unit_testing.html)
  + [URI Class](http://docs.google.com/uri.html)
  + [User Agent Class](http://docs.google.com/user_agent.html)
  + [XML-RPC and XML-RPC Server Classes](http://docs.google.com/xmlrpc.html)
  + [Zip Encoding Class](http://docs.google.com/zip.html)
* [Database Reference](http://docs.google.com/database/index.html)
  + [Quick Start: Usage Examples](http://docs.google.com/database/examples.html)
  + [Database Configuration](http://docs.google.com/database/configuration.html)
  + [Connecting to a Database](http://docs.google.com/database/connecting.html)
  + [Running Queries](http://docs.google.com/database/queries.html)
  + [Generating Query Results](http://docs.google.com/database/results.html)
  + [Query Helper Functions](http://docs.google.com/database/helpers.html)
  + [Query Builder Class](http://docs.google.com/database/query_builder.html)
  + [Transactions](http://docs.google.com/database/transactions.html)
  + [Getting MetaData](http://docs.google.com/database/metadata.html)
  + [Custom Function Calls](http://docs.google.com/database/call_function.html)
  + [Query Caching](http://docs.google.com/database/caching.html)
  + [Database Manipulation with Database Forge](http://docs.google.com/database/forge.html)
  + [Database Utilities Class](http://docs.google.com/database/utilities.html)
  + [Database Driver Reference](http://docs.google.com/database/db_driver_reference.html)
* [Helpers](http://docs.google.com/helpers/index.html)
  + [Array Helper](http://docs.google.com/helpers/array_helper.html)
  + [CAPTCHA Helper](http://docs.google.com/helpers/captcha_helper.html)
  + [Cookie Helper](http://docs.google.com/helpers/cookie_helper.html)
  + [Date Helper](http://docs.google.com/helpers/date_helper.html)
  + [Directory Helper](http://docs.google.com/helpers/directory_helper.html)
  + [Download Helper](http://docs.google.com/helpers/download_helper.html)
  + [Email Helper](http://docs.google.com/helpers/email_helper.html)
  + [File Helper](http://docs.google.com/helpers/file_helper.html)
  + [Form Helper](http://docs.google.com/helpers/form_helper.html)
  + [HTML Helper](http://docs.google.com/helpers/html_helper.html)
  + [Inflector Helper](http://docs.google.com/helpers/inflector_helper.html)
  + [Language Helper](http://docs.google.com/helpers/language_helper.html)
  + [Number Helper](http://docs.google.com/helpers/number_helper.html)
  + [Path Helper](http://docs.google.com/helpers/path_helper.html)
  + [Security Helper](http://docs.google.com/helpers/security_helper.html)
  + [Smiley Helper](http://docs.google.com/helpers/smiley_helper.html)
  + [String Helper](http://docs.google.com/helpers/string_helper.html)
  + [Text Helper](http://docs.google.com/helpers/text_helper.html)
  + [Typography Helper](http://docs.google.com/helpers/typography_helper.html)
  + [URL Helper](http://docs.google.com/helpers/url_helper.html)
  + [XML Helper](http://docs.google.com/helpers/xml_helper.html)

Toggle Table of Contents

[CodeIgniter](http://docs.google.com/index.html)

* [Welcome to CodeIgniter](http://docs.google.com/general/welcome.html)
* [Installation Instructions](http://docs.google.com/installation/index.html)
  + [Downloading CodeIgniter](http://docs.google.com/installation/downloads.html)
  + [Installation Instructions](http://docs.google.com/installation/index.html)
  + [Upgrading From a Previous Version](http://docs.google.com/installation/upgrading.html)
  + [Troubleshooting](http://docs.google.com/installation/troubleshooting.html)
* [CodeIgniter Overview](http://docs.google.com/overview/index.html)
  + [Getting Started](http://docs.google.com/overview/getting_started.html)
  + [CodeIgniter at a Glance](http://docs.google.com/overview/at_a_glance.html)
  + [Supported Features](http://docs.google.com/overview/features.html)
  + [Application Flow Chart](http://docs.google.com/overview/appflow.html)
  + [Model-View-Controller](http://docs.google.com/overview/mvc.html)
  + [Architectural Goals](http://docs.google.com/overview/goals.html)
* [Tutorial](http://docs.google.com/tutorial/index.html)
  + [Static pages](http://docs.google.com/tutorial/static_pages.html)
  + [News section](http://docs.google.com/tutorial/news_section.html)
  + [Create news items](http://docs.google.com/tutorial/create_news_items.html)
  + [Conclusion](http://docs.google.com/tutorial/conclusion.html)
* [Contributing to CodeIgniter](http://docs.google.com/contributing/index.html)
  + [Writing CodeIgniter Documentation](http://docs.google.com/documentation/index.html)
  + [Developer’s Certificate of Origin 1.1](http://docs.google.com/DCO.html)
* [General Topics](http://docs.google.com/general/index.html)
  + [CodeIgniter URLs](http://docs.google.com/general/urls.html)
  + [Controllers](http://docs.google.com/general/controllers.html)
  + [Reserved Names](http://docs.google.com/general/reserved_names.html)
  + [Views](http://docs.google.com/general/views.html)
  + [Models](http://docs.google.com/general/models.html)
  + [Helpers](http://docs.google.com/general/helpers.html)
  + [Using CodeIgniter Libraries](http://docs.google.com/general/libraries.html)
  + [Creating Libraries](http://docs.google.com/general/creating_libraries.html)
  + [Using CodeIgniter Drivers](http://docs.google.com/general/drivers.html)
  + [Creating Drivers](http://docs.google.com/general/creating_drivers.html)
  + [Creating Core System Classes](http://docs.google.com/general/core_classes.html)
  + [Creating Ancillary Classes](http://docs.google.com/general/ancillary_classes.html)
  + [Hooks - Extending the Framework Core](http://docs.google.com/general/hooks.html)
  + [Auto-loading Resources](http://docs.google.com/general/autoloader.html)
  + [Common Functions](http://docs.google.com/general/common_functions.html)
  + [Compatibility Functions](http://docs.google.com/general/compatibility_functions.html)
  + [URI Routing](http://docs.google.com/general/routing.html)
  + [Error Handling](http://docs.google.com/general/errors.html)
  + [Caching](http://docs.google.com/general/caching.html)
  + [Profiling Your Application](http://docs.google.com/general/profiling.html)
  + [Running via the CLI](http://docs.google.com/general/cli.html)
  + [Managing your Applications](http://docs.google.com/general/managing_apps.html)
  + [Handling Multiple Environments](http://docs.google.com/general/environments.html)
  + [Alternate PHP Syntax for View Files](http://docs.google.com/general/alternative_php.html)
  + [Security](http://docs.google.com/general/security.html)
  + [PHP Style Guide](http://docs.google.com/general/styleguide.html)
* [Libraries](http://docs.google.com/index.html)
  + [Benchmarking Class](http://docs.google.com/benchmark.html)
  + [Caching Driver](http://docs.google.com/caching.html)
  + [Calendaring Class](http://docs.google.com/calendar.html)
  + [Shopping Cart Class](http://docs.google.com/cart.html)
  + [Config Class](http://docs.google.com/config.html)
  + [Email Class](http://docs.google.com/email.html)
  + [Encrypt Class](http://docs.google.com/encrypt.html)
  + [Encryption Library](#gjdgxs)
  + [File Uploading Class](http://docs.google.com/file_uploading.html)
  + [Form Validation](http://docs.google.com/form_validation.html)
  + [FTP Class](http://docs.google.com/ftp.html)
  + [Image Manipulation Class](http://docs.google.com/image_lib.html)
  + [Input Class](http://docs.google.com/input.html)
  + [Javascript Class](http://docs.google.com/javascript.html)
  + [Language Class](http://docs.google.com/language.html)
  + [Loader Class](http://docs.google.com/loader.html)
  + [Migrations Class](http://docs.google.com/migration.html)
  + [Output Class](http://docs.google.com/output.html)
  + [Pagination Class](http://docs.google.com/pagination.html)
  + [Template Parser Class](http://docs.google.com/parser.html)
  + [Security Class](http://docs.google.com/security.html)
  + [Session Library](http://docs.google.com/sessions.html)
  + [HTML Table Class](http://docs.google.com/table.html)
  + [Trackback Class](http://docs.google.com/trackback.html)
  + [Typography Class](http://docs.google.com/typography.html)
  + [Unit Testing Class](http://docs.google.com/unit_testing.html)
  + [URI Class](http://docs.google.com/uri.html)
  + [User Agent Class](http://docs.google.com/user_agent.html)
  + [XML-RPC and XML-RPC Server Classes](http://docs.google.com/xmlrpc.html)
  + [Zip Encoding Class](http://docs.google.com/zip.html)
* [Database Reference](http://docs.google.com/database/index.html)
  + [Quick Start: Usage Examples](http://docs.google.com/database/examples.html)
  + [Database Configuration](http://docs.google.com/database/configuration.html)
  + [Connecting to a Database](http://docs.google.com/database/connecting.html)
  + [Running Queries](http://docs.google.com/database/queries.html)
  + [Generating Query Results](http://docs.google.com/database/results.html)
  + [Query Helper Functions](http://docs.google.com/database/helpers.html)
  + [Query Builder Class](http://docs.google.com/database/query_builder.html)
  + [Transactions](http://docs.google.com/database/transactions.html)
  + [Getting MetaData](http://docs.google.com/database/metadata.html)
  + [Custom Function Calls](http://docs.google.com/database/call_function.html)
  + [Query Caching](http://docs.google.com/database/caching.html)
  + [Database Manipulation with Database Forge](http://docs.google.com/database/forge.html)
  + [Database Utilities Class](http://docs.google.com/database/utilities.html)
  + [Database Driver Reference](http://docs.google.com/database/db_driver_reference.html)
* [Helpers](http://docs.google.com/helpers/index.html)
  + [Array Helper](http://docs.google.com/helpers/array_helper.html)
  + [CAPTCHA Helper](http://docs.google.com/helpers/captcha_helper.html)
  + [Cookie Helper](http://docs.google.com/helpers/cookie_helper.html)
  + [Date Helper](http://docs.google.com/helpers/date_helper.html)
  + [Directory Helper](http://docs.google.com/helpers/directory_helper.html)
  + [Download Helper](http://docs.google.com/helpers/download_helper.html)
  + [Email Helper](http://docs.google.com/helpers/email_helper.html)
  + [File Helper](http://docs.google.com/helpers/file_helper.html)
  + [Form Helper](http://docs.google.com/helpers/form_helper.html)
  + [HTML Helper](http://docs.google.com/helpers/html_helper.html)
  + [Inflector Helper](http://docs.google.com/helpers/inflector_helper.html)
  + [Language Helper](http://docs.google.com/helpers/language_helper.html)
  + [Number Helper](http://docs.google.com/helpers/number_helper.html)
  + [Path Helper](http://docs.google.com/helpers/path_helper.html)
  + [Security Helper](http://docs.google.com/helpers/security_helper.html)
  + [Smiley Helper](http://docs.google.com/helpers/smiley_helper.html)
  + [String Helper](http://docs.google.com/helpers/string_helper.html)
  + [Text Helper](http://docs.google.com/helpers/text_helper.html)
  + [Typography Helper](http://docs.google.com/helpers/typography_helper.html)
  + [URL Helper](http://docs.google.com/helpers/url_helper.html)
  + [XML Helper](http://docs.google.com/helpers/xml_helper.html)

[CodeIgniter](http://docs.google.com/index.html)

* [Docs](http://docs.google.com/index.html) »
* [Libraries](http://docs.google.com/index.html) »
* Encryption Library

classic layout

Encryption Library[**¶**](#30j0zll)

Important

DO NOT use this or any other *encryption* library for user password storage! Passwords must be *hashed* instead, and you should do that via PHP’s own [Password Hashing extension](http://php.net/password).

The Encryption Library provides two-way data encryption. To do so in a cryptographically secure way, it utilizes PHP extensions that are unfortunately not always available on all systems. You must meet one of the following dependencies in order to use this library:

* [OpenSSL](http://php.net/openssl)
* [MCrypt](http://php.net/mcrypt) (and MCRYPT\_DEV\_URANDOM availability)

If neither of the above dependencies is met, we simply cannot offer you a good enough implementation to meet the high standards required for proper cryptography.

* [Using the Encryption Library](#1fob9te)
  + [Initializing the Class](#3znysh7)
  + [Default behavior](#2et92p0)
  + [Setting your encryption\_key](#tyjcwt)
  + [Supported encryption ciphers and modes](#3dy6vkm)
    - [Portable ciphers](#1t3h5sf)
    - [Driver-specific ciphers](#4d34og8)
    - [Encryption modes](#2s8eyo1)
  + [Message Length](#17dp8vu)
  + [Configuring the library](#3rdcrjn)
  + [Encrypting and decrypting data](#26in1rg)
    - [How it works](#lnxbz9)
    - [Using custom parameters](#35nkun2)
    - [Supported HMAC authentication algorithms](#1ksv4uv)
* [Class Reference](#44sinio)

[**Using the Encryption Library**](#2jxsxqh)[**¶**](#1fob9te)

[**Initializing the Class**](#z337ya)[**¶**](#3znysh7)

Like most other classes in CodeIgniter, the Encryption library is initialized in your controller using the $this->load->library() method:

$this->load->library('encryption');

Once loaded, the Encryption library object will be available using:

$this->encryption

[**Default behavior**](#3j2qqm3)[**¶**](#2et92p0)

By default, the Encryption Library will use the AES-128 cipher in CBC mode, using your configured *encryption\_key* and SHA512 HMAC authentication.

Note

AES-128 is chosen both because it is proven to be strong and because of its wide availability across different cryptographic software and programming languages’ APIs.

However, the *encryption\_key* is not used as is.

If you are somewhat familiar with cryptography, you should already know that a HMAC also requires a secret key and using the same key for both encryption and authentication is a bad practice.

Because of that, two separate keys are derived from your already configured *encryption\_key*: one for encryption and one for authentication. This is done via a technique called [HMAC-based Key Derivation Function](http://en.wikipedia.org/wiki/HKDF) (HKDF).

[**Setting your encryption\_key**](#1y810tw)[**¶**](#tyjcwt)

An *encryption key* is a piece of information that controls the cryptographic process and permits a plain-text string to be encrypted, and afterwards - decrypted. It is the secret “ingredient” in the whole process that allows you to be the only one who is able to decrypt data that you’ve decided to hide from the eyes of the public. After one key is used to encrypt data, that same key provides the **only** means to decrypt it, so not only must you chose one carefully, but you must not lose it or you will also lose access to the data.

It must be noted that to ensure maximum security, such key *should* not only be as strong as possible, but also often changed. Such behavior however is rarely practical or possible to implement, and that is why CodeIgniter gives you the ability to configure a single key that is to be used (almost) every time.

It goes without saying that you should guard your key carefully. Should someone gain access to your key, the data will be easily decrypted. If your server is not totally under your control it’s impossible to ensure key security so you may want to think carefully before using it for anything that requires high security, like storing credit card numbers.

Your encryption key **must** be as long as the encyption algorithm in use allows. For AES-128, that’s 128 bits or 16 bytes (charcters) long. You will find a table below that shows the supported key lengths of different ciphers.

The key should be as random as possible and it **must not** be a regular text string, nor the output of a hashing function, etc. In order to create a proper key, you must use the Encryption library’s create\_key() method

// $key will be assigned a 16-byte (128-bit) random key  
$key = $this->encryption->create\_key(16);

The key can be either stored in your *application/config/config.php*, or you can design your own storage mechanism and pass the key dynamically when encrypting/decrypting.

To save your key to your *application/config/config.php*, open the file and set:

$config['encryption\_key'] = 'YOUR KEY';

You’ll notice that the create\_key() method outputs binary data, which is hard to deal with (i.e. a copy-paste may damage it), so you may use bin2hex(), hex2bin() or Base64-encoding to work with the key in a more friendly manner. For example:

// Get a hex-encoded representation of the key:  
$key = bin2hex($this->encryption->create\_key(16));  
  
// Put the same value in your config with hex2bin(),  
// so that it is still passed as binary to the library:  
$config['encryption\_key'] = hex2bin(<your hex-encoded key>);

[**Supported encryption ciphers and modes**](#4i7ojhp)[**¶**](#3dy6vkm)

Note

The terms ‘cipher’ and ‘encryption algorithm’ are interchangeable.

[**Portable ciphers**](#2xcytpi)[**¶**](#1t3h5sf)

Because MCrypt and OpenSSL (also called drivers throughout this document) each support different sets of encryption algorithms and often implement them in different ways, our Encryption library is designed to use them in a portable fashion, or in other words - it enables you to use them interchangeably, at least for the ciphers supported by both drivers.

It is also implemented in a way that aims to match the standard implementations in other programming languages and libraries.

Here’s a list of the so called “portable” ciphers, where “CodeIgniter name” is the string value that you’d have to pass to the Encryption library to use that cipher:

|  |  |  |  |
| --- | --- | --- | --- |
| Cipher name | CodeIgniter name | Key lengths (bits / bytes) | Supported modes |
| AES-128 / Rijndael-128 | aes-128 | 128 / 16 | CBC, CTR, CFB, CFB8, OFB, ECB |
| AES-192 | aes-192 | 192 / 24 | CBC, CTR, CFB, CFB8, OFB, ECB |
| AES-256 | aes-256 | 256 / 32 | CBC, CTR, CFB, CFB8, OFB, ECB |
| DES | des | 56 / 7 | CBC, CFB, CFB8, OFB, ECB |
| TripleDES | tripledes | 56 / 7, 112 / 14, 168 / 21 | CBC, CFB, CFB8, OFB |
| Blowfish | blowfish | 128-448 / 16-56 | CBC, CFB, OFB, ECB |
| CAST5 / CAST-128 | cast5 | 88-128 / 11-16 | CBC, CFB, OFB, ECB |
| RC4 / ARCFour | rc4 | 40-2048 / 5-256 | Stream |

Important

Because of how MCrypt works, if you fail to provide a key with the appropriate length, you might end up using a different algorithm than the one configured, so be really careful with that!

Note

In case it isn’t clear from the above table, Blowfish, CAST5 and RC4 support variable length keys. That is, any number in the shown ranges is valid, although in bit terms that only happens in 8-bit increments.

Note

Even though CAST5 supports key lengths lower than 128 bits (16 bytes), in fact they will just be zero-padded to the maximum length, as specified in [RFC 2144](http://tools.ietf.org/rfc/rfc2144.txt).

Note

Blowfish supports key lengths as small as 32 bits (4 bytes), but our tests have shown that only lengths of 128 bits (16 bytes) or higher are properly supported by both MCrypt and OpenSSL. It is also a bad practice to use such low-length keys anyway.

[**Driver-specific ciphers**](#1ci93xb)[**¶**](#4d34og8)

As noted above, MCrypt and OpenSSL support different sets of encryption ciphers. For portability reasons and because we haven’t tested them properly, we do not advise you to use the ones that are driver-specific, but regardless, here’s a list of most of them:

|  |  |  |  |
| --- | --- | --- | --- |
| Cipher name | Driver | Key lengths (bits / bytes) | Supported modes |
| AES-128 | OpenSSL | 128 / 16 | CBC, CTR, CFB, CFB8, OFB, ECB, XTS |
| AES-192 | OpenSSL | 192 / 24 | CBC, CTR, CFB, CFB8, OFB, ECB, XTS |
| AES-256 | OpenSSL | 256 / 32 | CBC, CTR, CFB, CFB8, OFB, ECB, XTS |
| Rijndael-128 | MCrypt | 128 / 16, 192 / 24, 256 / 32 | CBC, CTR, CFB, CFB8, OFB, OFB8, ECB |
| Rijndael-192 | MCrypt | 128 / 16, 192 / 24, 256 / 32 | CBC, CTR, CFB, CFB8, OFB, OFB8, ECB |
| Rijndael-256 | MCrypt | 128 / 16, 192 / 24, 256 / 32 | CBC, CTR, CFB, CFB8, OFB, OFB8, ECB |
| GOST | MCrypt | 256 / 32 | CBC, CTR, CFB, CFB8, OFB, OFB8, ECB |
| Twofish | MCrypt | 128 / 16, 192 / 24, 256 / 32 | CBC, CTR, CFB, CFB8, OFB, OFB8, ECB |
| CAST-128 | MCrypt | 40-128 / 5-16 | CBC, CTR, CFB, CFB8, OFB, OFB8, ECB |
| CAST-256 | MCrypt | 128 / 16, 192 / 24, 256 / 32 | CBC, CTR, CFB, CFB8, OFB, OFB8, ECB |
| Loki97 | MCrypt | 128 / 16, 192 / 24, 256 / 32 | CBC, CTR, CFB, CFB8, OFB, OFB8, ECB |
| SaferPlus | MCrypt | 128 / 16, 192 / 24, 256 / 32 | CBC, CTR, CFB, CFB8, OFB, OFB8, ECB |
| Serpent | MCrypt | 128 / 16, 192 / 24, 256 / 32 | CBC, CTR, CFB, CFB8, OFB, OFB8, ECB |
| XTEA | MCrypt | 128 / 16 | CBC, CTR, CFB, CFB8, OFB, OFB8, ECB |
| RC2 | MCrypt | 8-1024 / 1-128 | CBC, CTR, CFB, CFB8, OFB, OFB8, ECB |
| RC2 | OpenSSL | 8-1024 / 1-128 | CBC, CFB, OFB, ECB |
| Camellia-128 | OpenSSL | 128 / 16 | CBC, CFB, CFB8, OFB, ECB |
| Camellia-192 | OpenSSL | 192 / 24 | CBC, CFB, CFB8, OFB, ECB |
| Camellia-256 | OpenSSL | 256 / 32 | CBC, CFB, CFB8, OFB, ECB |
| Seed | OpenSSL | 128 / 16 | CBC, CFB, OFB, ECB |

Note

If you wish to use one of those ciphers, you’d have to pass its name in lower-case to the Encryption library.

Note

You’ve probably noticed that all AES cipers (and Rijndael-128) are also listed in the portable ciphers list. This is because drivers support different modes for these ciphers. Also, it is important to note that AES-128 and Rijndael-128 are actually the same cipher, but **only** when used with a 128-bit key.

Note

CAST-128 / CAST-5 is also listed in both the portable and driver-specific ciphers list. This is because OpenSSL’s implementation doesn’t appear to be working correctly with key sizes of 80 bits and lower.

Note

RC2 is listed as supported by both MCrypt and OpenSSL. However, both drivers implement them differently and they are not portable. It is probably worth noting that we only found one obscure source confirming that it is MCrypt that is not properly implementing it.

[**Encryption modes**](#3whwml4)[**¶**](#2s8eyo1)

Different modes of encryption have different characteristics and serve for different purposes. Some are stronger than others, some are faster and some offer extra features. We are not going in depth into that here, we’ll leave that to the cryptography experts. The table below is to provide brief informational reference to our more experienced users. If you are a beginner, just stick to the CBC mode - it is widely accepted as strong and secure for general purposes.

|  |  |  |  |
| --- | --- | --- | --- |
| Mode name | CodeIgniter name | Driver support | Additional info |
| CBC | cbc | MCrypt, OpenSSL | A safe default choice |
| CTR | ctr | MCrypt, OpenSSL | Considered as theoretically better than CBC, but not as widely available |
| CFB | cfb | MCrypt, OpenSSL | N/A |
| CFB8 | cfb8 | MCrypt, OpenSSL | Same as CFB, but operates in 8-bit mode (not recommended). |
| OFB | ofb | MCrypt, OpenSSL | N/A |
| OFB8 | ofb8 | MCrypt | Same as OFB, but operates in 8-bit mode (not recommended). |
| ECB | ecb | MCrypt, OpenSSL | Ignores IV (not recommended). |
| XTS | xts | OpenSSL | Usually used for encrypting random access data such as RAM or hard-disk storage. |
| Stream | stream | MCrypt, OpenSSL | This is not actually a mode, it just says that a stream cipher is being used. Required because of the general cipher+mode initialization process. |

[**Message Length**](#2bn6wsx)[**¶**](#17dp8vu)

It’s probably important for you to know that an encrypted string is usually longer than the original, plain-text string (depending on the cipher).

This is influenced by the cipher algorithm itself, the IV prepended to the cipher-text and the HMAC authentication message that is also prepended. Furthermore, the encrypted message is also Base64-encoded so that it is safe for storage and transmission, regardless of a possible character set in use.

Keep this information in mind when selecting your data storage mechanism. Cookies, for example, can only hold 4K of information.

[**Configuring the library**](#qsh70q)[**¶**](#3rdcrjn)

For usability, performance, but also historical reasons tied to our old [Encrypt Class](http://docs.google.com/encrypt.html), the Encryption library is designed to use repeatedly the same driver, encryption cipher, mode and key.

As noted in the “Default behavior” section above, this means using an auto-detected driver (OpenSSL has a higher priority), the AES-128 ciper in CBC mode, and your $config['encryption\_key'] value.

If you wish to change that however, you need to use the initialize() method. It accepts an associative array of parameters, all of which are optional:

|  |  |
| --- | --- |
| Option | Possible values |
| driver | ‘mcrypt’, ‘openssl’ |
| cipher | Cipher name (see [Supported encryption ciphers and modes](#3as4poj)) |
| mode | Encryption mode (see [Encryption modes](#2s8eyo1)) |
| key | Encryption key |

For example, if you were to change the encryption algorithm and mode to AES-256 in CTR mode, this is what you should do:

$this->encryption->initialize(  
 array(  
 'cipher' => 'aes-256',  
 'mode' => 'ctr',  
 'key' => '<a 32-character random string>'  
 )  
);

Note that we only mentioned that you want to change the ciper and mode, but we also included a key in the example. As previously noted, it is important that you choose a key with a proper size for the used algorithm.

There’s also the ability to change the driver, if for some reason you have both, but want to use MCrypt instead of OpenSSL:

// Switch to the MCrypt driver  
$this->encryption->initialize(array('driver' => 'mcrypt'));  
  
// Switch back to the OpenSSL driver  
$this->encryption->initialize(array('driver' => 'openssl'));

[**Encrypting and decrypting data**](#1pxezwc)[**¶**](#26in1rg)

Encrypting and decrypting data with the already configured library settings is simple. As simple as just passing the string to the encrypt() and/or decrypt() methods:

$plain\_text = 'This is a plain-text message!';  
$ciphertext = $this->encryption->encrypt($plain\_text);  
  
// Outputs: This is a plain-text message!  
echo $this->encryption->decrypt($ciphertext);

And that’s it! The Encryption library will do everything necessary for the whole process to be cryptographically secure out-of-the-box. You don’t need to worry about it.

Important

Both methods will return FALSE in case of an error. While for encrypt() this can only mean incorrect configuration, you should always check the return value of decrypt() in production code.

[**How it works**](#49x2ik5)[**¶**](#lnxbz9)

If you must know how the process works, here’s what happens under the hood:

* $this->encryption->encrypt($plain\_text)
  1. Derive an encryption key and a HMAC key from your configured *encryption\_key* via HKDF, using the SHA-512 digest algorithm.
  2. Generate a random initialization vector (IV).
  3. Encrypt the data via AES-128 in CBC mode (or another previously configured cipher and mode), using the above-mentioned derived encryption key and IV.
  4. Prepend said IV to the resulting cipher-text.
  5. Base64-encode the resulting string, so that it can be safely stored or transferred without worrying about character sets.
  6. Create a SHA-512 HMAC authentication message using the derived HMAC key to ensure data integrity and prepend it to the Base64 string.
* $this->encryption->decrypt($ciphertext)
  1. Derive an encryption key and a HMAC key from your configured *encryption\_key* via HKDF, using the SHA-512 digest algorithm. Because your configured *encryption\_key* is the same, this will produce the same result as in the encrypt() method above - otherwise you won’t be able to decrypt it.
  2. Check if the string is long enough, separate the HMAC out of it and validate if it is correct (this is done in a way that prevents timing attacks against it). Return FALSE if either of the checks fails.
  3. Base64-decode the string.
  4. Separate the IV out of the cipher-text and decrypt the said cipher-text using that IV and the derived encryption key.

[**Using custom parameters**](#2p2csry)[**¶**](#35nkun2)

Let’s say you have to interact with another system that is out of your control and uses another method to encrypt data. A method that will most certainly not match the above-described sequence and probably not use all of the steps either.

The Encryption library allows you to change how its encryption and decryption processes work, so that you can easily tailor a custom solution for such situations.

Note

It is possible to use the library in this way, without setting an *encryption\_key* in your configuration file.

All you have to do is to pass an associative array with a few parameters to either the encrypt() or decrypt() method. Here’s an example:

// Assume that we have $ciphertext, $key and $hmac\_key  
// from on outside source  
  
$message = $this->encryption->decrypt(  
 $ciphertext,  
 array(  
 'cipher' => 'blowfish',  
 'mode' => 'cbc',  
 'key' => $key,  
 'hmac\_digest' => 'sha256',  
 'hmac\_key' => $hmac\_key  
 )  
);

In the above example, we are decrypting a message that was encrypted using the Blowfish cipher in CBC mode and authenticated via a SHA-256 HMAC.

Important

Note that both ‘key’ and ‘hmac\_key’ are used in this example. When using custom parameters, encryption and HMAC keys are not derived like the default behavior of the library is.

Below is a list of the available options.

However, unless you really need to and you know what you are doing, we advise you to not change the encryption process as this could impact security, so please do so with caution.

|  |  |  |  |
| --- | --- | --- | --- |
| Option | Default value | Mandatory / Optional | Description |
| cipher | N/A | Yes | Encryption algorithm (see [Supported encryption ciphers and modes](#3as4poj)). |
| mode | N/A | Yes | Encryption mode (see [Encryption modes](#2s8eyo1)). |
| key | N/A | Yes | Encryption key. |
| hmac | TRUE | No | Whether to use a HMAC. Boolean. If set to FALSE, then *hmac\_digest* and *hmac\_key* will be ignored. |
| hmac\_digest | sha512 | No | HMAC message digest algorithm (see [Supported HMAC authentication algorithms](#147n2zr)). |
| hmac\_key | N/A | Yes, unless *hmac* is FALSE | HMAC key. |
| raw\_data | FALSE | No | Whether the cipher-text should be raw. Boolean. If set to TRUE, then Base64 encoding and decoding will not be performed and HMAC will not be a hexadecimal string. |

Important

encrypt() and decrypt() will return FALSE if a mandatory parameter is not provided or if a provided value is incorrect. This includes *hmac\_key*, unless *hmac* is set to FALSE.

[**Supported HMAC authentication algorithms**](#3o7alnk)[**¶**](#1ksv4uv)

For HMAC message authentication, the Encryption library supports usage of the SHA-2 family of algorithms:

|  |  |  |
| --- | --- | --- |
| Algorithm | Raw length (bytes) | Hex-encoded length (bytes) |
| sha512 | 64 | 128 |
| sha384 | 48 | 96 |
| sha256 | 32 | 64 |
| sha224 | 28 | 56 |

The reason for not including other popular algorithms, such as MD5 or SHA1 is that they are no longer considered secure enough and as such, we don’t want to encourage their usage. If you absolutely need to use them, it is easy to do so via PHP’s native [hash\_hmac()](http://php.net/manual/en/function.hash-hmac.php) function.

Stronger algorithms of course will be added in the future as they appear and become widely available.

[**Class Reference**](#23ckvvd)[**¶**](#44sinio)

*class* CI\_Encryption[¶](#ihv636) initialize(*$params*)[¶](#32hioqz)

|  |  |
| --- | --- |
| Parameters: | * **$params** (*array*) – Configuration parameters |
| Returns: | CI\_Encryption instance (method chaining) |
| Return type: | CI\_Encryption |

Initializes (configures) the library to use a different driver, cipher, mode or key.

Example:

$this->encryption->initialize(  
 array('mode' => 'ctr')  
);

Please refer to the [Configuring the library](#1hmsyys) section for detailed info.

encrypt(*$data*[, *$params = NULL*])[¶](#41mghml)

|  |  |
| --- | --- |
| Parameters: | * **$data** (*string*) – Data to encrypt * **$params** (*array*) – Optional parameters |
| Returns: | Encrypted data or FALSE on failure |
| Return type: | string |

Encrypts the input data and returns its ciphertext.

Example:

$ciphertext = $this->encryption->encrypt('My secret message');

Please refer to the [Using custom parameters](#2grqrue) section for information on the optional parameters.

decrypt(*$data*[, *$params = NULL*])[¶](#vx1227)

|  |  |
| --- | --- |
| Parameters: | * **$data** (*string*) – Data to decrypt * **$params** (*array*) – Optional parameters |
| Returns: | Decrypted data or FALSE on failure |
| Return type: | string |

Decrypts the input data and returns it in plain-text.

Example:

echo $this->encryption->decrypt($ciphertext);

Please refer to the [Using custom parameters](#2grqrue) secrion for information on the optional parameters.

create\_key(*$length*)[¶](#3fwokq0)

|  |  |
| --- | --- |
| Parameters: | * **$length** (*int*) – Output length |
| Returns: | A pseudo-random cryptographic key with the specified length, or FALSE on failure |
| Return type: | string |

Creates a cryptographic key by fetching random data from the operating system’s sources (i.e. /dev/urandom).

hkdf(*$key*[, *$digest = 'sha512'*[, *$salt = NULL*[, *$length = NULL*[, *$info = ''*]]]])[¶](#1v1yuxt)

|  |  |
| --- | --- |
| Parameters: | * **$key** (*string*) – Input key material * **$digest** (*string*) – A SHA-2 family digest algorithm * **$salt** (*string*) – Optional salt * **$length** (*int*) – Optional output length * **$info** (*string*) – Optional context/application-specific info |
| Returns: | A pseudo-random key or FALSE on failure |
| Return type: | string |

Derives a key from another, presumably weaker key.

This method is used internally to derive an encryption and HMAC key from your configured *encryption\_key*.

It is publicly available due to its otherwise general purpose. It is described in [RFC 5869](https://tools.ietf.org/rfc/rfc5869.txt).

However, as opposed to the description in RFC 5869, this implementation doesn’t support SHA1.

Example:

$hmac\_key = $this->encryption->hkdf(  
 $key,  
 'sha512',  
 NULL,  
 NULL,  
 'authentication'  
);  
  
// $hmac\_key is a pseudo-random key with a length of 64 bytes

[Next](http://docs.google.com/file_uploading.html)   [Previous](http://docs.google.com/encrypt.html)

© Copyright 2014 - 2018, British Columbia Institute of Technology. Last updated on Jun 12, 2018.

Built with [Sphinx](http://sphinx-doc.org/) using a [theme](https://github.com/snide/sphinx_rtd_theme) provided by [Read the Docs](https://readthedocs.org).