Mcrypt

Introduction

This is an interface to the mcrypt library, which supports a wide variety of block algorithms such as DES, TripleDES, Blowfish (default), 3-WAY, SAFER-SK64, SAFER-SK128, TWOFISH, TEA, RC2 and GOST in CBC, OFB, CFB and ECB cipher modes. Additionally, it supports RC6 and IDEA which are considered "non-free".

Installing/Configuring

Requirements

These functions work using » mcrypt. To use it, download *libmcrypt-x.x.tar.gz* from » http://mcrypt.sourceforge.net/ and follow the included installation instructions. Windows users will find all the needed compiled mcrypt binaries at » http://files.edin.dk/php/win32/mcrypt/.

As of PHP 5.0.0 you will need libmcrypt Version 2.5.6 or greater.

If you linked against libmcrypt 2.4.x or higher, the following additional block algorithms are supported: CAST, LOKI97, RIJNDAEL, SAFERPLUS, SERPENT and the following stream ciphers: ENIGMA (crypt), PANAMA, RC4 and WAKE. With libmcrypt 2.4.x or higher another cipher mode is also available; nOFB.

Installation

You need to compile PHP with the *--with-mcrypt[=DIR]* parameter to enable this extension. DIR is the mcrypt install directory. Make sure you compile libmcrypt with the option *--disable-posix-threads*.

Runtime Configuration

The behaviour of these functions is affected by settings in *php.ini*.

Mcrypt configuration options

Name	Default	Changeable	Changelog
mcrypt.algorithms_dir	NULL	PHP_INI_ALL	Available since PHP 4.0.2.
mcrypt.modes_dir	NULL	PHP_INI_ALL	Available since PHP 4.0.2.

For further details and definitions of the PHP_INI_* constants, see the php.ini directives.

Resource Types

mcrypt_module_open() returns an encryption descriptor.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

Mcrypt can operate in four block cipher modes (CBC, OFB, CFB, and ECB). If linked against libmcrypt-2.4.x or higher the functions can also operate in the block cipher mode nOFB and in STREAM mode. Below you find a list with all supported encryption modes together with the constants that are defines for the encryption mode. For a more complete reference and discussion see Applied Cryptography by Schneier (ISBN 0-471-11709-9).

- MCRYPT_MODE_ECB (electronic codebook) is suitable for random data, such as encrypting other keys. Since data there is short and random, the disadvantages of ECB have a favorable negative effect.
- MCRYPT_MODE_CBC (cipher block chaining) is especially suitable for encrypting files where the security is increased over ECB significantly.
- MCRYPT_MODE_CFB (cipher feedback) is the best mode for encrypting byte streams where single bytes must be encrypted.
- MCRYPT_MODE_OFB (output feedback, in 8bit) is comparable to CFB, but can be used in applications where error propagation cannot be tolerated. It's insecure (because it operates in 8bit mode) so it is not recommended to use it.
- MCRYPT_MODE_NOFB (output feedback, in nbit) is comparable to OFB, but more secure because it operates on the block size of the algorithm.
- MCRYPT_MODE_STREAM is an extra mode to include some stream algorithms like WAKE or RC4.

Some other mode and random device constants:
MCRYPT_ENCRYPT (integer)

MCRYPT_DECRYPT (integer)

MCRYPT_DEV_RANDOM (integer)

MCRYPT_DEV_URANDOM (integer)

MCRYPT_RAND (integer)

Mcrypt ciphers

Here is a list of ciphers which are currently supported by the mcrypt extension. For a complete list of supported ciphers, see the defines at the end of *mcrypt.h*. The general rule with the mcrypt-2.2.x API is that you can access the cipher from PHP with MCRYPT_ciphername. With the libmcrypt-2.4.x and libmcrypt-2.5.x API these constants also work, but it is possible to specify the name of the cipher as a string with a call to mcrypt_module_open().

- MCRYPT_3DES
- MCRYPT_ARCFOUR_IV (libmcrypt > 2.4.x only)
- MCRYPT_ARCFOUR (libmcrypt > 2.4.x only)
- MCRYPT_BLOWFISH
- MCRYPT_CAST_128
- MCRYPT_CAST_256
- MCRYPT_CRYPT
- MCRYPT_DES
- MCRYPT_DES_COMPAT (libmcrypt 2.2.x only)
- MCRYPT_ENIGMA (libmcrypt > 2.4.x only, alias for MCRYPT_CRYPT)
- MCRYPT_GOST
- MCRYPT_IDEA (non-free)
- MCRYPT_LOKI97 (libmcrypt > 2.4.x only)
- MCRYPT_MARS (libmcrypt > 2.4.x only, non-free)
- MCRYPT_PANAMA (libmcrypt > 2.4.x only)
- MCRYPT_RIJNDAEL_128 (libmcrypt > 2.4.x only)
- MCRYPT_RIJNDAEL_192 (libmcrypt > 2.4.x only)
- MCRYPT_RIJNDAEL_256 (libmcrypt > 2.4.x only)
- MCRYPT_RC2
- MCRYPT_RC4 (libmcrypt 2.2.x only)
- MCRYPT_RC6 (libmcrypt > 2.4.x only)
- MCRYPT_RC6_128 (libmcrypt 2.2.x only)
- MCRYPT_RC6_192 (libmcrypt 2.2.x only)
- MCRYPT_RC6_256 (libmcrypt 2.2.x only)
- MCRYPT_SAFER64
- MCRYPT_SAFER128
- MCRYPT_SAFERPLUS (libmcrypt > 2.4.x only)

- MCRYPT_SERPENT(libmcrypt > 2.4.x only)
- MCRYPT_SERPENT_128 (libmcrypt 2.2.x only)
- MCRYPT_SERPENT_192 (libmcrypt 2.2.x only)
- MCRYPT_SERPENT_256 (libmcrypt 2.2.x only)
- MCRYPT_SKIPJACK (libmcrypt > 2.4.x only)
- MCRYPT_TEAN (libmcrypt 2.2.x only)
- MCRYPT_THREEWAY
- MCRYPT_TRIPLEDES (libmcrypt > 2.4.x only)
- MCRYPT_TWOFISH (for older mcrypt 2.x versions, or mcrypt > 2.4.x)
- MCRYPT_TWOFISH128 (TWOFISHxxx are available in newer 2.x versions, but not in the 2.4.x versions)
- MCRYPT_TWOFISH192
- MCRYPT_TWOFISH256
- MCRYPT_WAKE (libmcrypt > 2.4.x only)
- MCRYPT_XTEA (libmcrypt > 2.4.x only)

You must (in CFB and OFB mode) or can (in CBC mode) supply an initialization vector (IV) to the respective cipher function. The IV must be unique and must be the same when decrypting/encrypting. With data which is stored encrypted, you can take the output of a function of the index under which the data is stored (e.g. the MD5 key of the filename). Alternatively, you can transmit the IV together with the encrypted data (see chapter 9.3 of Applied Cryptography by Schneier (ISBN 0-471-11709-9) for a discussion of this topic).

Examples

Mcrypt can be used to encrypt and decrypt using the above mentioned ciphers. If you linked against libmcrypt-2.2.x, the four important mcrypt commands (mcrypt_cfb(), mcrypt_cbc(), mcrypt_ecb(), and mcrypt_ofb()) can operate in both modes which are named MCRYPT_ENCRYPT and MCRYPT_DECRYPT, respectively.

Example #1 - Encrypt an input value with TripleDES under 2.2.x in ECB mode

```
<?php
$key = "this is a secret key";
$input = "Let us meet at 9 o'clock at the secret place.";

$encrypted_data = mcrypt_ecb (MCRYPT_3DES, $key, $input, MCRYPT_ENCRYPT);
?>
```

This example will give you the encrypted data as a string in *$encrypted_data*.

If you linked against libmcrypt 2.4.x or 2.5.x, these functions are still available, but it is recommended that you use the advanced functions.

Example #2 - Encrypt an input value with TripleDES under 2.4.x and higher in ECB mode

```
<?php
   $key = "this is a secret key";
   $input = "Let us meet at 9 o'clock at the secret place.";

   $td = mcrypt_module_open('tripledes', '', 'ecb', '');
   $iv = mcrypt_create_iv (mcrypt_enc_get_iv_size($td), MCRYPT_RAND);
   mcrypt_generic_init($td, $key, $iv);
   $encrypted_data = mcrypt_generic($td, $input);
   mcrypt_generic_deinit($td);
   mcrypt_module_close($td);
?>
```

This example will give you the encrypted data as a string in *$encrypted_data*. For a full example see mcrypt_module_open().

Mcrypt Functions

mcrypt_cbc

mcrypt_cbc -- Encrypt/decrypt data in CBC mode

Description

string **mcrypt_cbc** (int `$cipher`, string `$key`, string `$data`, int `$mode` [, string `$iv`])

string **mcrypt_cbc** (string `$cipher`, string `$key`, string `$data`, int `$mode` [, string `$iv`])

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or higher. The *`mode`* should be either **MCRYPT_ENCRYPT** or **MCRYPT_DECRYPT**.

This function should not be used anymore, see mcrypt_generic() and mdecrypt_generic() for replacements.

mcrypt_cfb

mcrypt_cfb -- Encrypt/decrypt data in CFB mode

Description

string **mcrypt_cfb** (int `$cipher`, string `$key`, string `$data`, int `$mode`, string `$iv`)

string **mcrypt_cfb** (string `$cipher`, string `$key`, string `$data`, int `$mode` [, string `$iv`])

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or higher. The *`mode`* should be either **MCRYPT_ENCRYPT** or **MCRYPT_DECRYPT**.

This function should not be used anymore, see mcrypt_generic() and mdecrypt_generic() for replacements.

mcrypt_create_iv

mcrypt_create_iv -- Create an initialization vector (IV) from a random source

Description

string **mcrypt_create_iv** (int `$size` [, int `$source`])

mcrypt_create_iv() is used to create an IV.

Parameter *`size`* determines the size of the IV, parameter *`source`* (defaults to random value) specifies the source of the IV.

The source can be **MCRYPT_RAND** (system random number generator), **MCRYPT_DEV_RANDOM** (read data from */dev/random*) and **MCRYPT_DEV_URANDOM** (read data from */dev/urandom*). **MCRYPT_RAND** is the only one supported on Windows because Windows (of course) doesn't have */dev/random* or */dev/urandom*.

Note

When using **MCRYPT_RAND**, remember to call srand() before mcrypt_create_iv() to initialize the random number generator; it is not seeded automatically like rand() is.

Example #3 - mcrypt_create_iv() example

```
<?php
   $size = mcrypt_get_iv_size(MCRYPT_CAST_256, MCRYPT_MODE_CFB);
   $iv = mcrypt_create_iv($size, MCRYPT_DEV_RANDOM);
?>
```

The IV is only meant to give an alternative seed to the encryption routines. This IV does not need to be secret at all, though it can be desirable. You even can send it along with your ciphertext without losing security.

More information can be found at » http://www.ciphersbyritter.com/GLOSSARY.HTM#IV, » http://fn2.freenet.edmonton.ab.ca/~jsavard/crypto/co0409.htm and in chapter 9.3 of Applied Cryptography by Schneier (ISBN 0-471-11709-9) for a discussion of this topic.

mcrypt_decrypt

mcrypt_decrypt -- Decrypts crypttext with given parameters

Description

string **mcrypt_decrypt** (string `$cipher`, string `$key`, string `$data`, string `$mode` [, string `$iv`])

Decrypts the *`data`* and returns the unencrypted data.

Parameters

`cipher`
: *`cipher`* is one of the MCRYPT_ciphername constants of the name of the algorithm as string.

`key`
: *`key`* is the key with which the data is encrypted. If it's smaller that the required keysize, it is padded with ' *\0* '.

`data`
: *`data`* is the data that will be decrypted with the given cipher and mode. If the size of the data is not n * blocksize, the data will be padded with ' *\0* '.

`mode`
: *`mode`* is one of the MCRYPT_MODE_modename constants of one of "ecb", "cbc", "cfb", "ofb", "nofb" or "stream".

`iv`
: The *`iv`* parameter is used for the initialisation in CBC, CFB, OFB modes, and in some algorithms in STREAM mode. If you do not supply an IV, while it is needed for an algorithm, the function issues a warning and uses an IV with all bytes set to ' *\0* '.

Return Values

Returns the decrypted data as a string.

mcrypt_ecb

mcrypt_ecb -- Deprecated: Encrypt/decrypt data in ECB mode

Description

string **mcrypt_ecb** (int `$cipher`, string `$key`, string `$data`, int `$mode`)

string **mcrypt_ecb** (string `$cipher`, string `$key`, string `$data`, int `$mode` [, string `$iv`])

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or higher. The *`mode`* should be either **MCRYPT_ENCRYPT** or **MCRYPT_DECRYPT**.

This function is deprecated and should not be used anymore, see mcrypt_generic() and mdecrypt_generic() for replacements.

mcrypt_enc_get_algorithms_name

mcrypt_enc_get_algorithms_name -- Returns the name of the opened algorithm

Description

string **mcrypt_enc_get_algorithms_name** (resource `$td`)

This function returns the name of the algorithm.

Examples

Example #4 - mcrypt_enc_get_algorithms_name() **example**

```
<?php
$td = mcrypt_module_open(MCRYPT_CAST_256, '', MCRYPT_MODE_CFB, '');
echo mcrypt_enc_get_algorithms_name($td). "\n";

$td = mcrypt_module_open('cast-256', '', MCRYPT_MODE_CFB, '');
echo mcrypt_enc_get_algorithms_name($td). "\n";
?>
```

The above example will output:

```
CAST-256
CAST-256
```

mcrypt_enc_get_block_size

mcrypt_enc_get_block_size -- Returns the blocksize of the opened algorithm

Description

int **mcrypt_enc_get_block_size** (resource `$td`)

Gets the blocksize of the opened algorithm.

Parameters

`td`
The encryption descriptor.

Return Values

Returns the block size of the specified algorithm in bytes.

mcrypt_enc_get_iv_size

mcrypt_enc_get_iv_size -- Returns the size of the IV of the opened algorithm

Description

int **mcrypt_enc_get_iv_size** (resource `$td`)

This function returns the size of the IV of the algorithm specified by the encryption descriptor in bytes. An IV is used in cbc, cfb and ofb modes, and in some algorithms in stream mode.

Parameters

`td`
The encryption descriptor.

Return Values

Returns the size of the IV, or 0 if the IV is ignored in the algorithm.

mcrypt_enc_get_key_size

mcrypt_enc_get_key_size -- Returns the maximum supported keysize of the opened mode

Description

int **mcrypt_enc_get_key_size** (resource `$td`)

Gets the maximum supported key size of the algorithm in bytes.

Parameters

`td`
The encryption descriptor.

Return Values

Returns the maximum supported key size of the algorithm in bytes.

mcrypt_enc_get_modes_name

mcrypt_enc_get_modes_name -- Returns the name of the opened mode

Description

string **mcrypt_enc_get_modes_name** (resource `$td`)

This function returns the name of the mode.

Parameters

`td`
The encryption descriptor.

Return Values

Returns the name as a string.

Examples

Example #5 - mcrypt_enc_get_modes_name() **example**

```
<?php
$td = mcrypt_module_open (MCRYPT_CAST_256, '', MCRYPT_MODE_CFB, '');
echo mcrypt_enc_get_modes_name($td). "\n";

$td = mcrypt_module_open ('cast-256', '', 'ecb', '');
echo mcrypt_enc_get_modes_name($td). "\n";
?>
```

The above example will output:

```
CFB
ECB
```

mcrypt_enc_get_supported_key_sizes

mcrypt_enc_get_supported_key_sizes -- Returns an array with the supported keysizes of the opened algorithm

Description

array **mcrypt_enc_get_supported_key_sizes** (resource `$td`)

Returns an array with the key sizes supported by the algorithm specified by the encryption descriptor. If it returns an empty array then all key sizes between 1 and mcrypt_enc_get_key_size() are supported by the algorithm.

Examples

Example #6 - mcrypt_enc_get_supported_key_sizes() example

```
<?php
   $td = mcrypt_module_open('rijndael-256', '', 'ecb', '');
   var_dump(mcrypt_enc_get_supported_key_sizes($td));
?>
```

This will print:

```
array(3) {
 [0]=>
 int(16)
 [1]=>
 int(24)
 [2]=>
 int(32)
}
```

mcrypt_enc_is_block_algorithm_mode

mcrypt_enc_is_block_algorithm_mode -- Checks whether the encryption of the opened mode works on blocks

Description

bool **mcrypt_enc_is_block_algorithm_mode** (resource `$td`)

Tells whether the algorithm of the opened mode works on blocks (e.g. **FALSE** for stream, and **TRUE** for cbc, cfb, ofb)..

Parameters

`td`
The encryption descriptor.

Return Values

Returns **TRUE** if the mode is for use with block algorithms, otherwise it returns **FALSE**.

mcrypt_enc_is_block_algorithm

mcrypt_enc_is_block_algorithm -- Checks whether the algorithm of the opened mode is a block algorithm

Description

bool **mcrypt_enc_is_block_algorithm** (resource `$td`)

Tells whether the algorithm of the opened mode is a block algorithm.

Parameters

`td`
The encryption descriptor.

Return Values

Returns **TRUE** if the algorithm is a block algorithm or **FALSE** if it is a stream one.

mcrypt_enc_is_block_mode

mcrypt_enc_is_block_mode -- Checks whether the opened mode outputs blocks

Description

bool **mcrypt_enc_is_block_mode** (resource `$td`)

Tells whether the opened mode outputs blocks (e.g. **TRUE** for cbc and ecb, and **FALSE** for cfb and stream).

Parameters

`td`
The encryption descriptor.

Return Values

Returns **TRUE** if the mode outputs blocks of bytes or **FALSE** if it outputs bytes.

mcrypt_enc_self_test

mcrypt_enc_self_test -- Runs a self test on the opened module

Description

int **mcrypt_enc_self_test** (resource `$td`)

This function runs the self test on the algorithm specified by the descriptor *`td`*.

Parameters

`td`
: The encryption descriptor.

Return Values

If the self test succeeds it returns **FALSE**. In case of an error, it returns **TRUE**.

mcrypt_encrypt

mcrypt_encrypt -- Encrypts plaintext with given parameters

Description

string **mcrypt_encrypt** (string `$cipher`, string `$key`, string `$data`, string `$mode` [, string `$iv`])

mcrypt_encrypt() encrypts the data and returns the encrypted data.

`Cipher` is one of the MCRYPT_ciphername constants of the name of the algorithm as string.

`Key` is the key with which the data will be encrypted. If it's smaller that the required keysize, it is padded with ' *\0* '. It is better not to use ASCII strings for keys. It is recommended to use the mhash functions to create a key from a string.

`Data` is the data that will be encrypted with the given cipher and mode. If the size of the data is not n * blocksize, the data will be padded with ' *\0* '. The returned crypttext can be larger that the size of the data that is given by *`data`*.

`Mode` is one of the MCRYPT_MODE_modename constants of one of "ecb", "cbc", "cfb", "ofb", "nofb" or "stream".

The *`IV`* parameter is used for the initialisation in CBC, CFB, OFB modes, and in some algorithms in STREAM mode. If you do not supply an IV, while it is needed for an algorithm, the function issues a warning and uses an IV with all bytes set to ' *\0* '.

Examples

Example #7 - mcrypt_encrypt() Example

```
<?php
   $iv_size = mcrypt_get_iv_size(MCRYPT_RIJNDAEL_256, MCRYPT_MODE_ECB);
   $iv = mcrypt_create_iv($iv_size, MCRYPT_RAND);
   $key = "This is a very secret key";
   $text = "Meet me at 11 o'clock behind the monument.";
   echo strlen($text) . "\n";

   $crypttext = mcrypt_encrypt(MCRYPT_RIJNDAEL_256, $key, $text,
MCRYPT_MODE_ECB, $iv);
   echo strlen($crypttext) . "\n";
?>
```

The above example will output:

```
42
64
```

See also mcrypt_module_open() for a more advanced API and an example.

mcrypt_generic_deinit

mcrypt_generic_deinit -- This function deinitializes an encryption module

Description

bool **mcrypt_generic_deinit** (resource `$td`)

This function terminates encryption specified by the encryption descriptor (*`td`*). It clears all buffers, but does not close the module. You need to call mcrypt_module_close() yourself. (But PHP does this for you at the end of the script.) Returns **FALSE** on error, or **TRUE** on success.

See for an example mcrypt_module_open() and the entry on mcrypt_generic_init().

mcrypt_generic_end

mcrypt_generic_end -- This function terminates encryption

Description

bool **mcrypt_generic_end** (resource `$td`)

Warning
This function is deprecated, use mcrypt_generic_deinit() instead. It can cause crashes when used with mcrypt_module_close() due to multiple buffer frees.

This function terminates encryption specified by the encryption descriptor (*`td`*). Actually it clears all buffers, and closes all the modules used. Returns **FALSE** on error, or **TRUE** on success.

mcrypt_generic_init

mcrypt_generic_init -- This function initializes all buffers needed for encryption

Description

int **mcrypt_generic_init** (resource `$td`, string `$key`, string `$iv`)

You need to call this function before every call to mcrypt_generic() or mdecrypt_generic().

Parameters

`td`
The encryption descriptor.

`key`
The maximum length of the key should be the one obtained by calling mcrypt_enc_get_key_size() and every value smaller than this is legal.

`iv`
The IV should normally have the size of the algorithms block size, but you must obtain the size by calling mcrypt_enc_get_iv_size(). IV is ignored in ECB. IV MUST exist in CFB, CBC, STREAM, nOFB and OFB modes. It needs to be random and unique (but not secret). The same IV must be used for encryption/decryption. If you do not want to use it you should set it to zeros, but this is not recommended.

Return Values

The function returns a negative value on error, -3 when the key length was incorrect, -4 when there was a memory allocation problem and any other return value is an unknown error. If an error occurs a warning will be displayed accordingly. **FALSE** is returned if incorrect parameters were passed.

See Also

- mcrypt_module_open()

mcrypt_generic

mcrypt_generic -- This function encrypts data

Description

string **mcrypt_generic** (resource `$td`, string `$data`)

This function encrypts data. The data is padded with " *\0* " to make sure the length of the data is n * blocksize. This function returns the encrypted data. Note that the length of the returned string can in fact be longer then the input, due to the padding of the data.

If you want to store the encrypted data in a database make sure to store the entire string as returned by mcrypt_generic, or the string will not entirely decrypt properly. If your original string is 10 characters long and the block size is 8 (use mcrypt_enc_get_block_size() to determine the blocksize), you would need at least 16 characters in your database field. Note the string returned by mdecrypt_generic() will be 16 characters as well...use rtrim() ($str, "\0") to remove the padding.

If you are for example storing the data in a MySQL database remember that varchar fields automatically have trailing spaces removed during insertion. As encrypted data can end in a space (ASCII 32), the data will be damaged by this removal. Store data in a tinyblob/tinytext (or larger) field instead.

The encryption handle should always be initialized with mcrypt_generic_init() with a key and an IV before calling this function. Where the encryption is done, you should free the encryption buffers by calling mcrypt_generic_deinit(). See mcrypt_module_open() for an example.

See also mdecrypt_generic(), mcrypt_generic_init(), and mcrypt_generic_deinit().

mcrypt_get_block_size

mcrypt_get_block_size -- Get the block size of the specified cipher

Description

int **mcrypt_get_block_size** (int `$cipher`)

int **mcrypt_get_block_size** (string `$cipher`, string `$module`)

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or 2.5.x.

mcrypt_get_block_size() is used to get the size of a block of the specified *`cipher`* (in combination with an encryption mode).

It is more useful to use the mcrypt_enc_get_block_size() function as this uses the resource returned by mcrypt_module_open().

This example shows how to use this function when linked against libmcrypt 2.4.x and 2.5.x.

Example #8 - mcrypt_get_block_size() example

```
<?php
   echo mcrypt_get_block_size('tripledes', 'ecb');
?>

Prints:
8
```

See also: mcrypt_get_key_size(), mcrypt_enc_get_block_size() and mcrypt_encrypt().

mcrypt_get_cipher_name

mcrypt_get_cipher_name -- Get the name of the specified cipher

Description

string **mcrypt_get_cipher_name** (int `$cipher`)

string **mcrypt_get_cipher_name** (string `$cipher`)

mcrypt_get_cipher_name() is used to get the name of the specified cipher.

mcrypt_get_cipher_name() takes the cipher number as an argument (libmcrypt 2.2.x) or takes the cipher name as an argument (libmcrypt 2.4.x or higher) and returns the name of the cipher or **FALSE**, if the cipher does not exist.

Examples

Example #9 - mcrypt_get_cipher_name() **Example**

```
<?php
  $cipher = MCRYPT_TripleDES;

  echo mcrypt_get_cipher_name($cipher);
?>
```

The above example will output:

```
3DES
```

mcrypt_get_iv_size

mcrypt_get_iv_size -- Returns the size of the IV belonging to a specific cipher/mode combination

Description

int **mcrypt_get_iv_size** (string `$cipher`, string `$mode`)

mcrypt_get_iv_size() returns the size of the Initialisation Vector (IV) in bytes. On error the function returns **FALSE**. If the IV is ignored in the specified cipher/mode combination zero is returned.

`cipher` is one of the MCRYPT_ciphername constants of the name of the algorithm as string.

`mode` is one of the MCRYPT_MODE_modename constants or one of "ecb", "cbc", "cfb", "ofb", "nofb" or "stream". The IV is ignored in ECB mode as this mode does not require it. You will need to have the same IV (think: starting point) both at encryption and decryption stages, otherwise your encryption will fail.

It is more useful to use the mcrypt_enc_get_iv_size() function as this uses the resource returned by mcrypt_module_open().

Examples

Example #10 - mcrypt_get_iv_size() example

```
<?php
   echo mcrypt_get_iv_size(MCRYPT_CAST_256, MCRYPT_MODE_CFB) . "\n";

   echo mcrypt_get_iv_size('des', 'ecb') . "\n";
?>
```

See also mcrypt_get_block_size(), mcrypt_enc_get_iv_size() and mcrypt_create_iv().

mcrypt_get_key_size

mcrypt_get_key_size -- Get the key size of the specified cipher

Description

int **mcrypt_get_key_size** (int `$cipher`)

int **mcrypt_get_key_size** (string `$cipher`, string `$module`)

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or 2.5.x.

mcrypt_get_key_size() is used to get the size of a key of the specified *`cipher`* (in combination with an encryption mode).

This example shows how to use this function when linked against libmcrypt 2.4.x and 2.5.x. It is more useful to use the mcrypt_enc_get_key_size() function as this uses the resource returned by mcrypt_module_open().

Example #11 - mcrypt_get_block_size() **example**

```
<?php
   echo mcrypt_get_key_size('tripledes', 'ecb');
?>

Prints:
24
```

See also: mcrypt_get_block_size(), **mcrypt_end_get_key_size()** and mcrypt_encrypt().

mcrypt_list_algorithms

mcrypt_list_algorithms -- Get an array of all supported ciphers

Description

array **mcrypt_list_algorithms** ([string `$lib_dir`])

mcrypt_list_algorithms() is used to get an array of all supported algorithms in the *`lib_dir`* parameter.

mcrypt_list_algorithms() takes an optional *`lib_dir`* parameter which specifies the directory where all algorithms are located. If not specifies, the value of the mcrypt.algorithms_dir *php.ini* directive is used.

Examples

Example #12 - mcrypt_list_algorithms() Example

```
<?php
   $algorithms = mcrypt_list_algorithms("/usr/local/lib/libmcrypt");

   foreach ($algorithms as $cipher) {
       echo "$cipher<br />\n";
   }
?>
```

The above example will produce a list with all supported algorithms in the "/usr/local/lib/libmcrypt" directory.

mcrypt_list_modes

mcrypt_list_modes -- Get an array of all supported modes

Description

array **mcrypt_list_modes** ([string `$lib_dir`])

mcrypt_list_modes() is used to get an array of all supported modes in the *`lib_dir`*.

mcrypt_list_modes() takes as optional parameter a directory which specifies the directory where all modes are located. If not specifies, the value of the mcrypt.modes_dir *php.ini* directive is used.

Examples

Example #13 - mcrypt_list_modes() Example

```
<?php
   $modes = mcrypt_list_modes();

   foreach ($modes as $mode) {
       echo "$mode <br />\n";
   }
?>
```

The above example will produce a list with all supported algorithms in the default mode directory. If it is not set with the ini directive mcrypt.modes_dir, the default directory of mcrypt is used (which is /usr/local/lib/libmcrypt).

mcrypt_module_close

mcrypt_module_close -- Close the mcrypt module

Description

bool **mcrypt_module_close** (resource `$td`)

Closes the specified encryption handle.

Parameters

`td`
The encryption descriptor.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- mcrypt_module_open()

mcrypt_module_get_algo_block_size

mcrypt_module_get_algo_block_size -- Returns the blocksize of the specified algorithm

Description

int **mcrypt_module_get_algo_block_size** (string `$algorithm` [, string `$lib_dir`])

Gets the blocksize of the specified algorithm.

Parameters

`algorithm`
: The algorithm name.

`lib_dir`
: This optional parameter can contain the location where the mode module is on the system.

Return Values

Returns the block size of the algorithm specified in bytes.

mcrypt_module_get_algo_key_size

mcrypt_module_get_algo_key_size -- Returns the maximum supported keysize of the opened mode

Description

int **mcrypt_module_get_algo_key_size** (string `$algorithm` [, string `$lib_dir`])

Gets the maximum supported keysize of the opened mode.

Parameters

`algorithm`
: The algorithm name.

`lib_dir`
: This optional parameter can contain the location where the mode module is on the system.

Return Values

This function returns the maximum supported key size of the algorithm specified in bytes.

mcrypt_module_get_supported_key_sizes

mcrypt_module_get_supported_key_sizes -- Returns an array with the supported keysizes of the opened algorithm

Description

array **mcrypt_module_get_supported_key_sizes** (string `$algorithm` [, string `$lib_dir`])

Returns an array with the key sizes supported by the specified algorithm. If it returns an empty array then all key sizes between 1 and mcrypt_module_get_algo_key_size() are supported by the algorithm. The optional *`lib_dir`* parameter can contain the location where the mode module is on the system.

See also mcrypt_enc_get_supported_key_sizes() which is used on open encryption modules.

mcrypt_module_is_block_algorithm_mode

mcrypt_module_is_block_algorithm_mode -- Returns if the specified module is a block algorithm or not

Description

bool **mcrypt_module_is_block_algorithm_mode** (string `$mode` [, string `$lib_dir`])

This function returns **TRUE** if the mode is for use with block algorithms, otherwise it returns **FALSE**. (e.g. **FALSE** for stream, and **TRUE** for cbc, cfb, ofb). The optional *`lib_dir`* parameter can contain the location where the mode module is on the system.

mcrypt_module_is_block_algorithm

mcrypt_module_is_block_algorithm -- This function checks whether the specified algorithm is a block algorithm

Description

bool **mcrypt_module_is_block_algorithm** (string `$algorithm` [, string `$lib_dir`])

This function returns **TRUE** if the specified algorithm is a block algorithm, or **FALSE** is it is a stream algorithm. The optional *`lib_dir`* parameter can contain the location where the algorithm module is on the system.

mcrypt_module_is_block_mode

mcrypt_module_is_block_mode -- Returns if the specified mode outputs blocks or not

Description

bool **mcrypt_module_is_block_mode** (string `$mode` [, string `$lib_dir`])

This function returns **TRUE** if the mode outputs blocks of bytes or **FALSE** if it outputs just bytes. (e.g. **TRUE** for cbc and ecb, and **FALSE** for cfb and stream). The optional *`lib_dir`* parameter can contain the location where the mode module is on the system.

mcrypt_module_open

mcrypt_module_open -- Opens the module of the algorithm and the mode to be used

Description

resource **mcrypt_module_open** (string `$algorithm`, string `$algorithm_directory`, string `$mode`, string `$mode_directory`)

This function opens the module of the algorithm and the mode to be used. The name of the algorithm is specified in algorithm, e.g. "twofish" or is one of the MCRYPT_ciphername constants. The module is closed by calling mcrypt_module_close(). Normally it returns an encryption descriptor, or **FALSE** on error.

The *`algorithm_directory`* and *`mode_directory`* are used to locate the encryption modules. When you supply a directory name, it is used. When you set one of these to the empty string (""), the value set by the *`mcrypt.algorithms_dir`* or *`mcrypt.modes_dir`* ini-directive is used. When these are not set, the default directories that are used are the ones that were compiled in into libmcrypt (usually /usr/local/lib/libmcrypt).

Examples

Example #14 - mcrypt_module_open() examples

```
<?php
   $td = mcrypt_module_open(MCRYPT_DES, '',
       MCRYPT_MODE_ECB, '/usr/lib/mcrypt-modes');

   $td = mcrypt_module_open('rijndael-256', '', 'ofb', '');
?>
```

The first line in the example above will try to open the DES cipher from the default directory and the EBC mode from the directory */usr/lib/mcrypt-modes*. The second example uses strings as name for the cipher and mode, this only works when the extension is linked against libmcrypt 2.4.x or 2.5.x.

Examples

Example #15 - Using mcrypt_module_open() in encryption

```
<?php
   /* Open the cipher */
   $td = mcrypt_module_open('rijndael-256', '', 'ofb', '');

   /* Create the IV and determine the keysize length, use MCRYPT_RAND
```

```
     * on Windows instead */
    $iv = mcrypt_create_iv(mcrypt_enc_get_iv_size($td), MCRYPT_DEV_RANDOM);
    $ks = mcrypt_enc_get_key_size($td);

    /* Create key */
    $key = substr(md5('very secret key'), 0, $ks);

    /* Intialize encryption */
    mcrypt_generic_init($td, $key, $iv);

    /* Encrypt data */
    $encrypted = mcrypt_generic($td, 'This is very important data');

    /* Terminate encryption handler */
    mcrypt_generic_deinit($td);

    /* Initialize encryption module for decryption */
    mcrypt_generic_init($td, $key, $iv);

    /* Decrypt encrypted string */
    $decrypted = mdecrypt_generic($td, $encrypted);

    /* Terminate decryption handle and close module */
    mcrypt_generic_deinit($td);
    mcrypt_module_close($td);

    /* Show string */
    echo trim($decrypted) . "\n";
?>
```

See also mcrypt_module_close(), mcrypt_generic(), mdecrypt_generic(), mcrypt_generic_init(), and mcrypt_generic_deinit().

mcrypt_module_self_test

mcrypt_module_self_test -- This function runs a self test on the specified module

Description

bool **mcrypt_module_self_test** (string `$algorithm` [, string `$lib_dir`])

This function runs the self test on the algorithm specified. The optional *`lib_dir`* parameter can contain the location of where the algorithm module is on the system.

The function returns **TRUE** if the self test succeeds, or **FALSE** when if fails.

Examples

Example #16 - mcrypt_module_self_test() **example**

```
<?php
var_dump(mcrypt_module_self_test(MCRYPT_RIJNDAEL_128)) . "\n";
var_dump(mcrypt_module_self_test(MCRYPT_BOGUS_CYPHER));
?>
```

The above example will output:

```
bool(true)
bool(false)
```

mcrypt_ofb

mcrypt_ofb -- Encrypt/decrypt data in OFB mode

Description

string **mcrypt_ofb** (int `$cipher`, string `$key`, string `$data`, int `$mode`, string `$iv`)

string **mcrypt_ofb** (string `$cipher`, string `$key`, string `$data`, int `$mode` [, string `$iv`])

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or higher. The *`mode`* should be either **MCRYPT_ENCRYPT** or **MCRYPT_DECRYPT**.

This function should not be used anymore, see mcrypt_generic() and mdecrypt_generic() for replacements.

mdecrypt_generic

mdecrypt_generic -- Decrypt data

Description

string **mdecrypt_generic** (resource `$td`, string `$data`)

This function decrypts data. Note that the length of the returned string can in fact be longer then the unencrypted string, due to the padding of the data.

Examples

Example #17 - mdecrypt_generic() example

```
<?php
   /* Data */
   $key = 'this is a very long key, even too long for the cipher';
   $plain_text = 'very important data';

   /* Open module, and create IV */
   $td = mcrypt_module_open('des', '', 'ecb', '');
   $key = substr($key, 0, mcrypt_enc_get_key_size($td));
   $iv_size = mcrypt_enc_get_iv_size($td);
   $iv = mcrypt_create_iv($iv_size, MCRYPT_RAND);

   /* Initialize encryption handle */
   if (mcrypt_generic_init($td, $key, $iv) != -1) {

       /* Encrypt data */
       $c_t = mcrypt_generic($td, $plain_text);
       mcrypt_generic_deinit($td);

       /* Reinitialize buffers for decryption */
       mcrypt_generic_init($td, $key, $iv);
       $p_t = mdecrypt_generic($td, $c_t);

       /* Clean up */
       mcrypt_generic_deinit($td);
       mcrypt_module_close($td);
   }

   if (strncmp($p_t, $plain_text, strlen($plain_text)) == 0) {
       echo "ok\n";
   } else {
       echo "error\n";
   }
?>
```

The above example shows how to check if the data before the encryption is the same as

the data after the decryption. It is very important to reinitialize the encryption buffer with mcrypt_generic_init() before you try to decrypt the data.

The decryption handle should always be initialized with mcrypt_generic_init() with a key and an IV before calling this function. Where the encryption is done, you should free the encryption buffers by calling mcrypt_generic_deinit(). See mcrypt_module_open() for an example.

See also mcrypt_generic(), mcrypt_generic_init(), and mcrypt_generic_deinit().