ZADKINE

leren
denken
durven
doen

Encryptie

1 maart 2016

Cryptography Extensions

- Crack — Cracklib
- CSPRNG
- Hash — HASH Message Digest Framework
- Mcrypt
- Mhash
- OpenSSL
- Password Hashing

Cryptography Extensions

- Crack — Cracklib

  These functions allow you to use the CrackLib library to test the 'strength' of a password. The 'strength' of a password is tested by that checks length, use of upper and lower case and checked against the specified CrackLib dictionary. CrackLib will also give helpful diagnostic messages that will help 'strengthen' the password.

- CSPRNG

  The » cryptographically secure pseudo-random number generator (CSPRNG) API provides an easy and reliable way to generate crypto-strong random integers and bytes for use within cryptographic contexts.
  This exists as of PHP 7.0.0 but there is also a » userland implementation for PHP >= 5.2.0.

Cryptography Extensions

- Hash — HASH Message Digest Framework

  Message Digest (hash) engine. Allows direct or incremental processing of arbitrary length messages using a variety of hashing algorithms.

- Mcrypt

  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". CFB/OFB are 8bit by default.

Cryptography Extensions

- Mhash

  These functions are intended to work with » mhash. Mhash can be used to create checksums, message digests, message authentication codes, and more.

  This is an interface to the mhash library. Mhash supports a wide variety of hash algorithms such as MD5, SHA1, GOST, and many others. For a complete list of supported hashes, refer to the constants page. The general rule is that you can access the hash algorithm from PHP with MHASH_hashname. For example, to access TIGER you use the PHP constant MHASH_TIGER.

- OpenSSL

  This module uses the functions of » OpenSSL for generation and verification of signatures and for sealing (encrypting) and opening (decrypting) data. OpenSSL offers many features that this module currently doesn't support. Some of these may be added in the future.

Cryptography Extensions

- Password Hashing

The password hashing API provides an easy to use wrapper around crypt() to make it easy to create and manage passwords in a secure manner.

This extension is available since PHP 5.5.0 but there is also an » userland implementation for PHP >= 5.3.7.

Mcrypt

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". CFB/OFB are 8bit by default.

Mcrypt library

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". CFB/OFB are 8bit by default.

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 defined for the encryption mode. For a more complete reference and discussion see Applied Cryptography by Schneier (ISBN 0-471-11709-9).

Predefined Constants

- **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".

Mcrypt ciphers

- 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)

ZADKINE

Mcrypt ciphers

- 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)

Example

Example #1

Encrypt an input value with AES with a 256-bit key under 2.4.x and higher in CBC mode

```
<?php
    $key = hash('sha256', 'this is a secret key', true);
    $input = "Let us meet at 9 o'clock at the secret place.";

    $td = mcrypt_module_open('rijndael-128', '', 'cbc', '');
    $iv = mcrypt_create_iv(mcrypt_enc_get_iv_size($td), MCRYPT_DEV_URANDOM);
    mcrypt_generic_init($td, $key, $iv);
    $encrypted_data = mcrypt_generic($td, $input);
    mcrypt_generic_deinit($td);
    mcrypt_module_close($td);
?>
```

Mcrypt Functions

mcrypt_cbc — Encrypts/decrypts data in CBC mode
mcrypt_cfb — Encrypts/decrypts data in CFB mode
mcrypt_create_iv — Creates an initialization vector (IV) from a random source
mcrypt_decrypt — Decrypts crypttext with given parameters
mcrypt_ecb — Deprecated: Encrypts/decrypts data in ECB mode
mcrypt_enc_get_algorithms_name — Returns the name of the opened algorithm
mcrypt_enc_get_block_size — Returns the blocksize of the opened algorithm
mcrypt_enc_get_iv_size — Returns the size of the IV of the opened algorithm
mcrypt_enc_get_key_size — Returns the maximum supported keysize of the opened mode
mcrypt_enc_get_modes_name — Returns the name of the opened mode
mcrypt_enc_get_supported_key_sizes — Returns an array with the supported keysizes of the opened algorithm
mcrypt_enc_is_block_algorithm_mode — Checks whether the encryption of the opened mode works on blocks
mcrypt_enc_is_block_algorithm — Checks whether the algorithm of the opened mode is a block algorithm
mcrypt_enc_is_block_mode — Checks whether the opened mode outputs blocks
mcrypt_enc_self_test — Runs a self test on the opened module

mcrypt_encrypt — Encrypts plaintext with given parameters

Mcrypt Functions

•mcrypt_generic_deinit — This function deinitializes an encryption module
•mcrypt_generic_end — This function terminates encryption
•mcrypt_generic_init — This function initializes all buffers needed for encryption
•mcrypt_generic — This function encrypts data
•mcrypt_get_block_size — Gets the block size of the specified cipher
•mcrypt_get_cipher_name — Gets the name of the specified cipher
•mcrypt_get_iv_size — Returns the size of the IV belonging to a specific cipher/mode combination
•mcrypt_get_key_size — Gets the key size of the specified cipher
•mcrypt_list_algorithms — Gets an array of all supported ciphers
•mcrypt_list_modes — Gets an array of all supported modes
•mcrypt_module_close — Closes the mcrypt module
•mcrypt_module_get_algo_block_size — Returns the blocksize of the specified algorithm
•mcrypt_module_get_algo_key_size — Returns the maximum supported keysize of the opened mode
•mcrypt_module_get_supported_key_sizes — Returns an array with the supported keysizes of the opened algorithm
•mcrypt_module_is_block_algorithm_mode — Returns if the specified module is a block algorithm or not
•mcrypt_module_is_block_algorithm — This function checks whether the specified algorithm is a block algorithm
•mcrypt_module_is_block_mode — Returns if the specified mode outputs blocks or not
•mcrypt_module_open — Opens the module of the algorithm and the mode to be used
•mcrypt_module_self_test — This function runs a self test on the specified module
•mcrypt_ofb — Encrypts/decrypts data in OFB mode
•mdecrypt_generic — Decrypts data

Example

```
function encrypt($sValue, $sSecretKey)
{
    return rtrim(
        base64_encode(
            mcrypt_encrypt(
                MCRYPT_RIJNDAEL_256,
                $sSecretKey, $sValue,
                MCRYPT_MODE_ECB,
                mcrypt_create_iv(
                    mcrypt_get_iv_size(
                        MCRYPT_RIJNDAEL_256,
                        MCRYPT_MODE_ECB
                    ),
                    MCRYPT_RAND)
                )
            ), "\0"
        );
}
```

ZADKINE

Example (cont)

```
function decrypt($sValue, $sSecretKey)
{
    return rtrim(
        mcrypt_decrypt(
            MCRYPT_RIJNDAEL_256,
            $sSecretKey,
            base64_decode($sValue),
            MCRYPT_MODE_ECB,
            mcrypt_create_iv(
                mcrypt_get_iv_size(
                    MCRYPT_RIJNDAEL_256,
                    MCRYPT_MODE_ECB
                ),
                MCRYPT_RAND
            )
        ), "\0"
    );

}
```