# [PHP, Simplest Two Way Encryption](http://stackoverflow.com/questions/9262109/php-simplest-two-way-encryption)

# Portable Data Encryption in PHP

If you're using [PHP 5.4 or newer](http://php.net/eol.php) and don't want to write a cryptography module yourself, I recommend using [an existing library that provides authenticated encryption](https://github.com/defuse/php-encryption). The library I linked relies only on what PHP provides and is under periodic review by a handful of security researchers. (Myself included.)

If your portability goals do not prevent requiring PECL extensions, [**libsodium**](https://pecl.php.net/package/libsodium) is highly recommended over anything you or I can write in PHP.

If you want to try your hand at cryptography engineering, read on.

First, you should take the time to learn [the dangers of unauthenticated encryption](https://paragonie.com/blog/2015/05/using-encryption-and-authentication-correctly#title.2.1) and [the Cryptographic Doom Principle](http://www.thoughtcrime.org/blog/the-cryptographic-doom-principle/).

* Encrypted data can still be tampered with by a malicious user.
* Authenticating the encrypted data prevents tampering.
* Authenticating the unencrypted data does not prevent tampering.

## Encryption and Decryption

Encryption in PHP is actually simple (we're going to use [openssl\_encrypt()](https://php.net/openssl_encrypt) and [openssl\_decrypt()](https://php.net/openssl_decrypt) once you have made some decisions about how to encrypt your information. Consult openssl\_get\_cipher\_methods() for a list of the methods supported on your system. Some good choices include [AES in CTR or CBC mode](http://www.daemonology.net/blog/2009-06-11-cryptographic-right-answers.html):

* aes-128-cbc
* aes-192-cbc
* aes-256-cbc
* aes-128-ctr
* aes-192-ctr
* aes-256-ctr

If you choose AES-CBC in openssl, it will default to PKCS7 padding (and strip it after decrypting), so you don't need to fuss over this detail. There is currently no reason to believe that the [AES key size](https://github.com/defuse/php-encryption/issues/35) is a significant issue to worry about (bigger is probably not better, due to bad key-scheduling in the 256-bit mode).

Note: **We are not using mcrypt because it is** [**abandonware**](http://sourceforge.net/projects/mcrypt/files/Libmcrypt) and has [unpatched bugs](http://sourceforge.net/p/mcrypt/patches/10) that might be security-affecting. Because of these reasons, I encourage other PHP developers to avoid it as well.

### Simple Encryption/Decryption Wrapper using OpenSSL

class UnsafeCrypto

{

const METHOD = 'aes-256-ctr';

/\*\*

\* Encrypts (but does not authenticate) a message

\*

\* @param string $message - plaintext message

\* @param string $key - encryption key (raw binary expected)

\* @param boolean $encode - set to TRUE to return a base64-encoded

\* @return string (raw binary)

\*/

public static function encrypt($message, $key, $encode = false)

{

$nonceSize = openssl\_cipher\_iv\_length(self::METHOD);

$nonce = openssl\_random\_pseudo\_bytes($nonceSize);

$ciphertext = openssl\_encrypt(

$message,

self::METHOD,

$key,

OPENSSL\_RAW\_DATA,

$nonce

);

// Now let's pack the IV and the ciphertext together

// Naively, we can just concatenate

if ($encode) {

return base64\_encode($nonce.$ciphertext);

}

return $nonce.$ciphertext;

}

/\*\*

\* Decrypts (but does not verify) a message

\*

\* @param string $message - ciphertext message

\* @param string $key - encryption key (raw binary expected)

\* @param boolean $encoded - are we expecting an encoded string?

\* @return string

\*/

public static function decrypt($message, $key, $encoded = false)

{

if ($encoded) {

$message = base64\_decode($message, true);

if ($message === false) {

throw new Exception('Encryption failure');

}

}

$nonceSize = openssl\_cipher\_iv\_length(self::METHOD);

$nonce = mb\_substr($message, 0, $nonceSize, '8bit');

$ciphertext = mb\_substr($message, $nonceSize, null, '8bit');

$plaintext = openssl\_decrypt(

$ciphertext,

self::METHOD,

$key,

OPENSSL\_RAW\_DATA,

$nonce

);

return $plaintext;

}

}

### Usage Example

$message = 'Ready your ammunition; we attack at dawn.';

$key = hex2bin('000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f');

$encrypted = UnsafeCrypto::encrypt($message, $key);

$decrypted = UnsafeCrypto::decrypt($encrypted, $key);

var\_dump($encrypted, $decrypted);

**Demo**: <https://3v4l.org/jl7qR>

This simple crypto library still is not safe to use. We need to authenticate ciphertexts and verify them before we decrypt.

**Note**: By default, UnsafeCrypto::encrypt() will return a raw binary string. Call it like this if you need to store it in a binary-safe format (base64-encoded):

$message = 'Ready your ammunition; we attack at dawn.';

$key = hex2bin('000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f');

$encrypted = UnsafeCrypto::encrypt($message, $key, true);

$decrypted = UnsafeCrypto::decrypt($encrypted, $key, true);

var\_dump($encrypted, $decrypted);

**Demo**: <http://3v4l.org/f5K93>

### Simple Authentication Wrapper

class SaferCrypto extends UnsafeCrypto

{

const HASH\_ALGO = 'sha256';

/\*\*

\* Encrypts then MACs a message

\*

\* @param string $message - plaintext message

\* @param string $key - encryption key (raw binary expected)

\* @param boolean $encode - set to TRUE to return a base64-encoded string

\* @return string (raw binary)

\*/

public static function encrypt($message, $key, $encode = false)

{

list($encKey, $authKey) = self::splitKeys($key);

// Pass to UnsafeCrypto::encrypt

$ciphertext = parent::encrypt($message, $encKey);

// Calculate a MAC of the IV and ciphertext

$mac = hash\_hmac(self::HASH\_ALGO, $ciphertext, $authKey, true);

if ($encode) {

return base64\_encode($mac.$ciphertext);

}

// Prepend MAC to the ciphertext and return to caller

return $mac.$ciphertext;

}

/\*\*

\* Decrypts a message (after verifying integrity)

\*

\* @param string $message - ciphertext message

\* @param string $key - encryption key (raw binary expected)

\* @param boolean $encoded - are we expecting an encoded string?

\* @return string (raw binary)

\*/

public static function decrypt($message, $key, $encoded = false)

{

list($encKey, $authKey) = self::splitKeys($key);

if ($encoded) {

$message = base64\_decode($message, true);

if ($message === false) {

throw new Exception('Encryption failure');

}

}

// Hash Size -- in case HASH\_ALGO is changed

$hs = mb\_strlen(hash(self::HASH\_ALGO, '', true), '8bit');

$mac = mb\_substr($message, 0, $hs, '8bit');

$ciphertext = mb\_substr($message, $hs, null, '8bit');

$calculated = hash\_hmac(

self::HASH\_ALGO,

$ciphertext,

$authKey,

true

);

if (!self::hashEquals($mac, $calculated)) {

throw new Exception('Encryption failure');

}

// Pass to UnsafeCrypto::decrypt

$plaintext = parent::decrypt($ciphertext, $encKey);

return $plaintext;

}

/\*\*

\* Splits a key into two separate keys; one for encryption

\* and the other for authenticaiton

\*

\* @param string $masterKey (raw binary)

\* @return array (two raw binary strings)

\*/

protected static function splitKeys($masterKey)

{

// You really want to implement HKDF here instead!

return [

hash\_hmac(self::HASH\_ALGO, 'ENCRYPTION', $masterKey, true),

hash\_hmac(self::HASH\_ALGO, 'AUTHENTICATION', $masterKey, true)

];

}

/\*\*

\* Compare two strings without leaking timing information

\*

\* @param string $a

\* @param string $b

\* @return boolean

\*/

protected static function hashEquals($a, $b)

{

if (function\_exists('hash\_equals')) {

return hash\_equals($a, $b);

}

$nonce = openssl\_random\_pseudo\_bytes(32);

return hash\_hmac(self::HASH\_ALGO, $a, $nonce) === hash\_hmac(self::HASH\_ALGO, $b, $nonce);

}

}

### Usage Example

$message = 'Ready your ammunition; we attack at dawn.';

$key = hex2bin('000102030405060708090a0b0c0d0e0f101112131415161718191a1b1c1d1e1f');

$encrypted = SaferCrypto::encrypt($message, $key);

$decrypted = SaferCrypto::decrypt($encrypted, $key);

var\_dump($encrypted, $decrypted);

**Examples**

<?php

include("../co.php");

include("config.php");

if($\_GET['del']!="" and is\_numeric($\_GET['del'])){

$user = addslashes($\_SESSION['user']);

$drop = "DELETE FROM notes WHERE id='".$\_GET['del']."' and username='".$user."'";

mysqli\_query($connect,$drop);

$\_SESSION['download']="0";

$\_SESSION['downloaded']="";

echo '<script>

$("#princ").html("Note deleted.");

$("#princ").load("../includes/note.php", function (responseText, textStatus, req) {

if (textStatus == "error") {

$("#princ").html("An error occurred");

}

});

setTimeout(function() {

$("#note").load("../includes/note.php?load=true", function (responseText, textStatus, req) {

if (textStatus == "error") {

$("#princ").html("An error occurred");

}

});

}, 10000);

</script>';

die();

}

if($\_POST['nota']!=""){//critta e inserisci nel db

function CasualPassword($lenght=527){

$available\_chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890";

$password = "";

for($i = 0; $i<$lenght; $i++){

$password .= substr($available\_chars,rand(0,strlen($available\_chars)-1),1);

}

return $password;

}

$key = CasualPassword();

$string = nl2br($\_POST['nota']);

for ($i = 0; $i < strlen($string); $i++ ) {

$temp = $string[$i] ^ $key[$i % strlen($key)];

$crypt .= str\_pad( dechex( ord( $temp ) ), 2, 0, STR\_PAD\_LEFT);

}

$encryptednote = $key.'\_\_'.$crypt;

$query = "INSERT INTO `notes` (

`username` ,

`nota`

)

VALUES (

'$user', '$encryptednote'

);";

mysqli\_query($connect,$query);

$\_SESSION['download']="0";

$\_SESSION['downloaded']="";

echo '<script>

$("#note").html("Loading...");

$("#note").load("../includes/note.php?load=true", function (responseText, textStatus, req) {

if (textStatus == "error") {

$("#princ").html("An error occurred.");

}

});

function update() {

$.get("../includes/note.php?load=true", function(data) {

$("#note").html(data);

});

}

window.setTimeout(update, 3000);

</script>';

}

else if($\_POST['nota']=="" and $\_GET['load']==""){echo '<script>

$(document).ready(function() {

$("#go").click(function(){

$("#note").html("Please wait...");

$.ajax({

url:"../includes/note.php",

type: "POST",

data: $("#notes").serialize(),

success: function(msg)

{

$("#note").html(msg);

},

error: function()

{

alert("Error!");

}

});

});

});

$("#note").html("Loading...");

$("#note").load("../includes/note.php?load=true", function (responseText, textStatus, req) {

if (textStatus == "error") {

$("#princ").html("An error occurred.");

}

});

function update() {

$.get("../includes/note.php?load=true", function(data) {

$("#note").html(data);

});

}

</script>

<form method="post" action="../includes/note.php" id="notes">

<textarea name="nota" id="nota" style="width:100%; height:20%;"></textarea>

<input type="button" id="go" value="Save">

</form>';}

if($\_GET['load']=="true"){echo '<h3>Your notes</h3>';

//seleziona tutte le note

$user = addslashes($\_SESSION['user']);

$download = addslashes($\_SESSION['download']);

$query = mysqli\_query($connect,"SELECT \* FROM notes WHERE username='".$user."' AND id>'".$download."' ORDER BY id DESC LIMIT 0,50");

while($note = mysqli\_fetch\_assoc($query)){

$separate = explode("\_\_",$encryptednote);

$key =$separate[0];

$crypt = $separate[1];

$cnt = 0;

for ($i = 0; $i < strlen( $crypt ); $i+=2){

$temp = chr( hexdec( substr( $crypt, $i, 2) ) );

$string .= $temp ^ $key[$cnt % strlen($key)];

$cnt++;

}

echo $string.'<br><button class="opzione" id="elimina'.$note['id'].'" onclick="$(\'#elimina'.$note['id'].'\').hide(); $(\'#confermaz'.$note['id'].'\').show();">Delete</button><button class="opzione" id="confermaz'.$note['id'].'" style="display:none;"><a href="javascript:apriLink(\'../includes/note.php?del='.$note['id'].'\')">Confirm</a></button><hr>';

if($note['id']>$\_SESSION['download'] or $\_SESSION['download']==""){$\_SESSION['download']=$note['id'];}

$\_SESSION['downloaded'] .= $stringa.'<br><button class="opzione" id="elimina'.$note['id'].'" onclick="$(\'#elimina'.$note['id'].'\').hide(); $(\'#confermaz'.$note['id'].'\').show();">Delete</button><button class="opzione" id="confermaz'.$note['id'].'" style="display:none;"><a href="javascript:apriLink(\'../includes/note.php?del='.$note['id'].'\')">Confirm</a></button><hr>';

$stringa="";

}

echo $\_SESSION['downloaded'];

echo '<script>

window.setTimeout(update, 3000); </script>';

die();

}

?>

<div id="note"></div>