What is the best way to encrypt a very short string in PHP?

Asked 16 years, 3 months ago Modified 10 years, 3 months ago

Viewed 12k times Part of PHP Collective

12

I would like to encrypt strings which could potentially only be about three or four characters but run to about twenty characters. A hashing function (md5, sha1, crypt etc) is not suitable as I would like to be able to decrypt the information as well. The mcrypt extension has a thoroughly daunting array of possibilities.

Does anyone have any ideas about the best way to safely encrypt short strings and *why*? Does anyone have any links to any material introducing a casual programmer to practical encryption scenarios?

PHP php encryption

Share

Improve this question

Follow

edited Sep 19, 2008 at 15:31

asked Sep 19, 2008 at 15:24

Shabbyrobe

12.6k ● 15 ● 62 ● 88

1 3-20 characters, this isn't a password, is it? One-way encryption is almost always a better choice for passwords. – James Socol Apr 30, 2009 at 17:31

8 Answers

Sorted by: Highest score (default)

7

I like to use GnuPG for anything that needs to be encrypted on a server and then possibly decrypted either on the server or on another server (which is usually my case). This allows for an extra level of security since in my scenario the encrypting server doesn't have the key to decrypt the data. It also allows for easier manual decryption. There are a few good wrappers available for various languages (another advantage), one for PHP is GnuPGP PHP Class.

Share Improve this answer

Follow

answered Sep 19, 2008 at 15:32

pdavis

3,211 ● 2 ● 31 ● 31

6

mcrypt is linked into most builds of PHP by default. It contains all the primitives you're likely to need. Without knowing more about what you're encrypting, what your threat model is, etc, it's hard to give concrete recommendations on what algorithm, mode of operation, etc to use.

One thing I can say for certain: With short text strings, it's more vital than ever that you MUST use a unique,

random Initialization Vector. Otherwise, it's trivial for someone to mount a variety of attacks against the encrypted data.

Share Improve this answer

Follow

answered Sep 19, 2008 at 16:50

Nick Johnson

101k ● 17 ● 130 ● 198

4

I highly recommend the suggestions of **Chris Kite**. Without knowing more about what you're doing, why, and the threats you anticipate needing to protect against AES-128 is likely sufficient. The ability to use symmetric encryption is great for a standalone app that will be both the decryptor and encryptor of data. As both **Chris Kite** and **Arachnid** said, due to the small size of your data it's advised that you pad the data and use a random Initialization Vector.

> **Update:** As for why.... if the data is small enough, and the IV can be predicted, it's possible to brute force the plain-text by generating cipher-text for every combination of plain-text with the known IV and matching it up to the captured cipher-text. In short, this is how rainbow tables work.

Now if you're going to encrypt on one server and decrypt on another I'd go with the suggestions of **pdavis**. By using an asymmetric method you're able to separate the encryption keys from the decryption keys. This way if the

server that encrypts data is compromised, the attacker is still unable to decrypt the data.

If you're able to, it'd help the community to know more about your use case for the encryption. As I mentioned above, having a proper understanding of plausible threats is key when evaluating security controls.

Share Improve this answer Follow

answered Sep 26, 2008 at 22:18

randy

356 ● 1 ● 4

2

I agree with Chris Kite - just use AES 128, this is far sufficient.

I don't know exactly your environment, but I guess you're transmitting the data somehow through the internet.

Don't use ECB, this will always produce the same result for the same plain text.

CBC mode is the way to go and don't forget a random initialization vector. This vector has to be communicated with the cipher text and can be sent in the clear.

Regarding your data, since AES is a block cipher, the outcome is always a multiple of the block size. If you don't want to let the observer know if your data is short or long, add some padding to extend it up to the maximum expected size.

Share Improve this answer

Follow

answered Sep 24, 2008 at 15:39

Oli
1,782 ● 3 ● 19 ● 22

2

If you want to encrypt and decrypt data within an application, you most likely want to use a symmetric key cipher. AES, which is the symmetric block encryption algorithm certified by the NSA for securing top secret data, is your best choice. There is a pure-PHP implementation available at www.phpaes.com

For your use it sounds like AES128 is sufficient. You will want to use CBC mode with a random initialization vector, or else the same data will always produce the same ciphertext.

Choosing the right encryption algorithm is a good first step, but there are many factors to a secure system which are hard to get right, such as key management. There are good resources out there, such as Applied Cryptography by Bruce Schneier, and Security Engineering by Ross Anderson (available for free online).

Share Improve this answer

Follow

edited Sep 26, 2008 at 17:21

answered Sep 19, 2008 at 16:35

Chris Kite
385 ● 1 ● 7

2

Does it matter if anybody can decrypt it? If you're just trying to obfuscate it *a little*, use ROT13. It's old school.

Share Improve this answer

Follow

edited Jun 23, 2014 at 15:42

franzlorenzon

5,933 ● 6 ● 37 ● 58

answered Sep 19, 2008 at 15:26

user1228

7 Did you honestly just suggest ROT13 in a crypto post? – Rich Bradshaw Sep 20, 2008 at 9:54

6 Holy crap, dude. Can you read? "Does it matter if anybody can decrypt it?" ROT13 is a valid encryption. Its trivial to break, but that doesn't necessarily matter always. I was clear enough about that, wasn't I? – user1228 Sep 22, 2008 at 13:22

0

Any one-way encryption algorithm such as Blowfish will do, I guess. Blowfish is fast and open. You can use Blowfish through the crypt() function. AFAIK there are no encryption algorithm that work especially well on small strings. One thing to be aware of though is that brute-forcing such small strings will be very easy. Maybe you should encrypt the string along with a 'secret' salt value for additional security.

Share Improve this answer

Follow

answered Sep 19, 2008 at 15:30

Roel

19.6k ● 7 ● 64 ● 97

Correction: the crypt() version isn't suitable because it doesn't have a way to decrypt. Use something like pear.php.net/package/Crypt_Blowfish. That'll give you the added advantage of not having to worry about installing an extra extension. – Roel Sep 19, 2008 at 15:34

0

You can use the general programming ideas without relying in built in encryption/decription functions Example create a function call it

```
function encryptstring($string) {

$string_length=strlen($string);
$encrychars="";
/**
*For each character of the given string generate the c
*/
for ($position = 0;$position<$string_length;$position+
    $key = (($string_length+$position)+1);
    $key = (255+$key) % 255;
    $get_char_to_be_encrypted = SUBSTR($string, $posit
    $ascii_char = ORD($get_char_to_be_encrypted);
    $xored_char = $ascii_char ^ $key;  //xor operation
    $encrypted_char = CHR($xored_char);
    $encrychars .= $encrypted_char;
}
/**
*Return the encrypted/decrypted string
*/
return $encrychars;
}
```

On the page with link to include the id's required to be encrypted

```
   /**
    *While passing the unique value to a link
    *Do the following steps
    */

    $id=57;//or if you are fetching it automatically j
     /**
     *For more security multiply some value
     *You can set the multiplication value in config f
     */
     $passstring=$id*346244;
     $encrypted_string=encryptstring($passstring);
     $param=urlencode($encrypted_string);
     /**
     *Derive the url for the link
     */
     echo '<a href="target_file.php?aZ98#9A_KL='.$para
```

On the target file that get opened after the link is clicked

```
     /**
      *Retriving the value in the target file
      *Do the following steps
      */
      $fetchid=$_GET['aZ98#9A_KL'];
      $passstring=urldecode(stripslashes($fetchid));
      $decrypted_string= encryptstring($passstring);
      /**
       *Divide the decrypted value with the same value
multiplication
       */
      $actual_id= $decrypted_string/346244;
```

Share Improve this answer

Follow

answered Sep 20, 2014 at 21:09

Nasz Njoka Sr.

1,118 ● 16 ● 28