**Using reCAPTCHA with PHP**

The reCAPTCHA PHP Library provides a simple way to place a [CAPTCHA](http://www.google.com/recaptcha/captcha) on your PHP website, helping you stop bots from abusing it. The library wraps the [reCAPTCHA API](http://code.google.com/apis/recaptcha/intro.html).

To use reCAPTCHA with PHP, you can download [reCAPTCHA PHP library](http://code.google.com/p/recaptcha/downloads/list?q=label:phplib-Latest). You will only need one file from there (recaptchalib.php). The other files are examples, readme and legal stuff -- they don't affect functionality.

**Quick Start**

After you've signed up for your API keys, below are basic instructions for installing reCAPTCHA on your site. A full reference guide to the PHP plugin can be found below.

**Client Side (How to make the CAPTCHA image show up)**

If you want to use the PHP library to display the reCAPTCHA widget, you'll need to insert this snippet of code inside the <form> element where the reCAPTCHA widget will be placed:

  require\_once('recaptchalib.php');  
  $publickey = "your\_public\_key"; // you got this from the signup page  
  echo recaptcha\_get\_html($publickey);

With the code, your form might look something like this:

  <html>  
    <body> <!-- the body tag is required or the CAPTCHA may not show on some browsers -->  
      <!-- your HTML content -->  
  
      <form method="post" action="verify.php">  
        <?php  
          require\_once('recaptchalib.php');  
          $publickey = "your\_public\_key"; // you got this from the signup page  
          echo recaptcha\_get\_html($publickey);  
        ?>  
        <input type="submit" />  
      </form>  
  
      <!-- more of your HTML content -->  
    </body>  
  </html>

Don't forget to set $publickey by replacing your\_public\_key with your API public key.

Note that the value of the "action" attribute is "verify.php". Now, verify.php is the destination file in which the values of this form are submitted to. So you will need a file verify.php in the same location as the client html.

The require\_once function in the example above expects recaptchalib.php to be in the same directory as your form file. If it is in another directory, you must link it appropriately. For example if your recaptchalib.php is in the directory called "captcha" that is on the same level as your form file, the function will look like this: require\_once('captcha/recaptchalib.php').

**Server Side (How to test if the user entered the right answer)**

The following code should be placed at the top of the verify.php file:

  <?php  
  require\_once('recaptchalib.php');  
  $privatekey = "your\_private\_key";  
  $resp = recaptcha\_check\_answer ($privatekey,  
                                $\_SERVER["REMOTE\_ADDR"],  
                                $\_POST["recaptcha\_challenge\_field"],  
                                $\_POST["recaptcha\_response\_field"]);  
  
  if (!$resp->is\_valid) {  
    // What happens when the CAPTCHA was entered incorrectly  
    die ("The reCAPTCHA wasn't entered correctly. Go back and try it again." .  
         "(reCAPTCHA said: " . $resp->error . ")");  
  } else {  
    // Your code here to handle a successful verification  
  }  
  ?>

In the code above:

 recaptcha\_check\_answer returns an object that represents whether the user successfully completed the challenge.

 If $resp->is\_valid is true then the captcha challenge was correctly completed and you should continue with form processing.

 If $resp->is\_valid is false then the user failed to provide the correct captcha text and you should redisplay the form to allow them another attempt. In this case $resp->error will be an error code that can be provided to recaptcha\_get\_html. Passing the error code makes the reCAPTCHA control display a message explaining that the user entered the text incorrectly and should try again.

Notice that this code is asking for the **private** key, which should not be confused with the public key. You get that from the same page as the public key.

Also make sure your form is set to get the form variables using $\_POST, instead of $\_REQUEST, and that the form itself is using the POST method.

That's it! reCAPTCHA should now be working on your site.

**Further Reading**

 [Customizing Look and Feel](http://code.google.com/apis/recaptcha/docs/customization.html)

 [Tips and Guidelines](http://code.google.com/apis/recaptcha/docs/tips.html)

 [Troubleshooting](http://code.google.com/apis/recaptcha/docs/troubleshooting.html)

**PHP Plugin Reference Guide**

Below is a comprehensive list of all the methods of the reCAPTCHA PHP Plugin.

**The recaptcha\_get\_html function**

The recaptcha\_get\_html function displays the HTML that presents reCAPTCHA to the user.

|  |  |
| --- | --- |
| recaptcha\_get\_html | |
| Parameter | |
| $pubkey -- string. required. | Your reCAPTCHA public key, from the [API Signup Page](https://www.google.com/recaptcha/admin/create?app=php) |
| $error -- string. optional (null is the default) | If this string is set, the reCAPTCHA area will display the error code given. This error code comes from ReCaptchaResponse->$error |
| $use\_ssl -- boolean. optional (false is default) | Should the SSL-based API be used? If you are displaying a page to the user over SSL, be sure to set this to true so an error dialog doesn't come up in the user's browser. |
| Return value | A string containing HTML to put on the web page. |

**The recaptcha\_check\_answer function**

After the user has filled out the HTML form, including their answer for the CAPTCHA, we want to check their answer when they submit the form using the recaptcha\_check\_answer function. The user's answer will be in two form fields, recaptcha\_challenge\_field and recaptcha\_response\_field. The reCAPTCHA library will make an HTTP request to the reCAPTCHA server and verify the user's answer.

|  |  |
| --- | --- |
| recaptcha\_check\_answer | |
| Parameter | |
| $privkey -- string. required. | Your reCAPTCHA private key, from the [API Signup Page](https://www.google.com/recaptcha/admin/create?app=php). |
| $remoteip -- string. required. | The user's IP address, in the format 192.168.0.1 |
| $challenge -- string. required. | The value of the form field recaptcha\_challenge\_field |
| $response -- string. required | The value of the form field recaptcha\_response\_field |
| Return value | An instance of the ReCaptchaResponse class |

|  |  |
| --- | --- |
| ReCaptchaResponse | |
| Field | |
| $is\_valid -- boolean | Did reCAPTCHA believe the answer was valid? |
| $error -- string | If the answer was invalid what was the problem? This error code can be used in recaptcha\_get\_html |
| Return value | The HTML or raw url to decode the email address, depending on which you function you called. |

**Mailhide**

The reCAPTCHA PHP Library includes bindings for the [Mailhide](http://www.google.com/recaptcha/mailhide/) API. This API allows you to wrap an email in a reCAPTCHA to prevent spammers from seeing it: exam[...](http://www.google.com/recaptcha/mailhide/d?k=01YVrDO6WGOpRXmUmxagsIkA==&c=rewEvxp1sUSNPrHRiX8quxJZ_HfM6j3L-tvj74kEFm8=)@example.com.

The Mailhide portion of the PHP Library requires the PHP [mcrypt](http://php.net/mcrypt) module.

The Mailhide API consists of two functions recaptcha\_mailhide\_html and recaptcha\_mailhide\_url. The functions have the same parameters. The \_html version returns HTML that can be directly put on your web page. The username portion of the email that is passed in is truncated and replaced with a link that calls Mailhide. The \_url version gives you the url to decode the email and leaves it up to you to place the email in HTML.

|  |  |
| --- | --- |
| recaptcha\_mailhide\_url / recaptcha\_mailhide\_html | |
| Parameter | |
| $pubkey -- string | The Mailhide public key from the [signup page](http://www.google.com/recaptcha/mailhide/apikey) |
| $privkey -- string | The Mailhide private key from the [signup page](http://www.google.com/recaptcha/mailhide/apikey) |
| $email -- string | The email address you want to hide. |

The following example shows how to use Mailhide:

<html><body>  
<?  
require\_once ("recaptchalib.php");  
// get a key at <http://www.google.com/recaptcha/mailhide/apikey>  
$mailhide\_pubkey = '';  
$mailhide\_privkey = '';  
?>  
The Mailhide encoding of example@example.com is  
<?  
echo recaptcha\_mailhide\_html ($mailhide\_pubkey,  
                              $mailhide\_privkey,  
                              "example@example.com");  
?>.  
<br>  
The url for the email is:  
<?  
echo recaptcha\_mailhide\_url ($mailhide\_pubkey,  
                             $mailhide\_privkey,  
                             "example@example.com");  
?>  
<br>  
</body></html>