

A no programming method to protected your web content

RMP Protected Content: Specification and application notes

(Draft version 0.3 April 2009)

1. Introduction

CarrotPay provides tools and services for web based merchants to enable them to cost effectively charge small amounts for their web content. Examples of such web content might be text, graphics, photographs, sounds, JavaScript and Flash games or combinations. This document describes how such content may be protected so that a user must pay each time the content is viewed or used.

2. Background

Web content is typically stored in a server's database or in simple files. For those people who have little or no programming experience, or who have limited access to server side tools, CarrotPay supports html only 'Buy Buttons' for simple files, which are easy to create and place on almost any web page (cut and paste is all that is required). A CarrotPay buy button hides the true URL of the content file until after full payment has been made, at which time the content's URL is returned to the user's web browser to be fetched and displayed. In this case the returned URL could be used many times to fetch the same content without further payment. This is often acceptable as the person has paid for the content, and could (if they wished), simply save the HTML page locally and view it multiple times. But what if that person passed the URL on to a friend (or worse, posted it onto a another web site). In this case their friend would also be able to view the content but without ever having to make a payment. RMP Protected Content is a method of stopping people from viewing content without paying each time, even if the URL is passed on by someone who has already paid.

There are various server side methods of protection such content but most of these methods are too 'heavy-weight' for small payments (e.g. US\$ 1.0 or less), as they involve the creation of sales records (or equivalent), with internal state which change as the transaction progresses. For small-valued content this may not be appropriate in terms of the programming effort, server resources and most importantly the time and effort required by the buyer. This document shows how content can easily be protected but without the need for any server-side support and no (or minimal), client side programming knowledge.

3. How it works

In simple terms, parts of your HTML page will be stored encrypted [using the Data Encryption Standard (DES)], and only decrypted after the correct authorisation code has been received as part of the result from a successful payment. This process does require some client side programming but CarrotPay has developed a JavaScript library (which hides the details from ordinary page developers), that binds together a payment request with content decryption in the web browser.

3.1 Web developer's prospective

There are three steps in deploying pages using `rpm:protectedContent`:

- 1) Register as a CarrotPay merchant (one time).
- 2) Follow the on-line wizard to create an `rpm:protectedContent` component for each product.
- 3) Include the `rpm` JavaScript library at the top of your page and paste the protected content component(s) into your web page. (See section 5 'Embedding RMP protected content into a Web page' for details).

NOTE: CarrotPay does not record details of your product and is not involved in product presentation or content hosting. CarrotPay's only responsibility is to receive Carrot-WebCoins on the merchant's behalf and to credit the merchant with the appropriate value and currency of the transaction.

3.2 Buyers prospective

There are three familiar steps to viewing rmp:protectedContent:

- 1) Visit any content protected page just for any normal html page.
- 2) Select the protected content with one click on a button or image etc.
- 3) Approve the payment (using their CarrotPurse), with zero or one click. The content will then be displayed in a few seconds.

NOTE: A buyer may make a payment even if they don't have available the merchant's base currency. The buyer will be presented with the item's price in their local currency and if approved, CarrotPay will automatically (and quickly), convert that currency into the merchant's base currency so that the merchant receives the exact amount expected.

4. The RMP protected content component

RMP protected content is declared in your HTML page using a new tag. The tag is called <rmp:protectedContent>. This tag requires a number of attributes and a body. The common way of creating this component is to use the CarrotPay Web Wizard provided through a Merchant Wallet. This wizard will lead you through a number of simple steps until all the rmp:protectedContent component data has been collected. You may then paste the generated component into the correct place in your web page (see section 5). For those merchants who have access to programming expertise or who prefer to store their product information in a database and generate their pages at runtime, rmp:protectedContent may be generated programmatically rather than by using the wizard (see section 6 for details).

Table 1. Attributes of the rmp:protectedContent element.

Attribute name	Description	Example
<i>service_url</i>	The url of the carrotPay service.	<i>http://www.carrot.org/payment</i>
<i>return_url</i>	Depending of your choice of 'inline' or file based content, this is either the site domain or domain plus path where the protected content file is stored.	<i>http://www.mysite.com/</i> <i>or</i> <i>http://www.mysite.com/content</i>
<i>fail_url</i>	The url of a page to be displayed if the payment is incomplete.	<i>http://www.mysite.com/fail.html</i>
<i>merchant_id</i>	The merchant's CarrotPay id.	<i>LBVJ-LWKV-JWCJ-RGSS</i>
<i>product_code</i>	This string should represent the product being sold. If 'inline' is NOT used, this attribute will be used as the filename part of the url of the protected content file.	<i>dog_on_bicycle</i> <i>or</i> <i>dog1234.txt</i>
<i>price</i>	The price to be charged in the merchant's registered currency.	<i>0.001, 1.0, 99.99</i>
<i>description</i>	A description of the product in a suitable language and UTF-8 encoded.	<i>"A dog on a bicycle"</i>
<i>id</i>	A string (unique for the target web page), identifying this particular protected content. This must be a valid HTML element id.	<i>pc_img_1</i>
<i>target_id</i>	A string (unique for the target web page), of an html tag where this content is to be displayed. This must be a	<i>showMeHere</i>

	valid HTML element id.	
<i>buy_button</i>	A string that specifies the url of a buy button icon. There are two special values that may be used "autosubmit" and "none". Autosubmit will cause the payment to be requested immediately the page is loaded. None will cause the payment to be requested only when activated by a trigger event.	http://images.carrotpay.com/btn120x45.gif or <i>autosubmit</i> or <i>none</i>
<i>trigger_id (optional)</i>	A string indicating which HTML tag an "onclick" event should be attached. This option is available only when buy_button is set to "none".	<i>img_01</i>

NOTE: The price and the product_code are both used in the encryption process and so they **must not** be changed once the content has been encrypted.

4.1 The online wizard

CarrotPay's online button wizard is normally used to create rmp:protectedContent components. The wizard has three main sections (Merchant, Product and Component), each containing a number of input fields. The wizard is normally started from a merchant wallet which will automatically fill all Merchant information fields.

The wizard may be initiated by using a merchant wallet and clicking on:

home->information->Start Web Wizard.

(A) Merchant information:

- i. **Merchant ID:** Example = "LBVJ-LWKV-JWCJ-RGSS"
- ii. **Currency:** Example = "USD"
- iii. **Hash Seed:** Example = "mjqpzpcjwxmbpzcb"

NOTE: A Merchant hash-seed should be kept secret at all times.

- iv. **Service URL:** Example = "http://www.carrot.org/payment"

NOTE: If you ever need to enter these fields by hand, the data is available in your CarrotPay Merchant wallet by clicking the 'Information' button on the 'Home' screen.

(B) Product Information

- i. **Product Code:** This is an identifier (usually unique), for a product on the merchant's site. If the content is not 'inline' (see later), this field will be used as the file name to retrieve the RMP Protected Content.
- ii. **Price:** The amount the merchant is asking for payment. It must be a valid price (in the range 0.001 to 999.999), and the currency is fixed for the specified Merchant ID.
- iii. **Content URL:** This URL will be used to retrieve the file where the merchant stores the RMP Protected Content file (if the content is not 'inline'). If the content is 'inline', the URL must be set to the web site's domain so that the buyer has a record of where they purchased the product.
- iv. **Fail URL:** This URL will be used to redirect the buyer if the purchase fails.
- v. **Description:** A brief description of the product in a language suitable for the buyer to read. The description must be UTF-8 encoded in order to be displayed correctly in the buyer's purse.

(C) Component information:

- i. **Component ID:** The id used for this component. If multiple components are used on a page each id must be unique within the page.
- ii. **Target ID:** The id of an html element (usually a <div> element), where the merchant wants the RMP Protected Content to be displayed. The same target id may be used for multiple rmp:protectedContent components, in which case the content will be replaced as each new content is paid for and decrypted.

NOTE: If the id is not known at the time the wizard is run, the merchant may use any temporary name (e.g. "xxx"), and then edit this by hand once the component is placed in an actual page.

- iii. **Trigger ID:** The id of an html element used to trigger the payment request. The component will automatically add an 'onclick' event to this element.

NOTES:

- This field is only used when the buy_button is set to "none".
 - If this optional attribute is **NOT** set, the merchant must add a payment trigger by themselves. (e.g. <div onclick="rmp.startPayment('pc_img_1');">)
- iv. **RMP Protected Content:** Any valid HTML code fragment that needs to be protected. This code typically includes text, tables, images, links and script elements.
 - v. **Inline option:** This is used to determine whether the protected content will be stored 'inline' (i.e. in the body of the rmp:protectedContent element), or in a file. If the RMP Protected Content is not inline, the content URL must be set appropriately so the URL can be used to retrieve the correct file.
 - vi. **CarrotPay Button options:** There are currently three options.
 - Use a CarrotPay buy button
In this case the payment process will be started when this button is clicked and you must choose the button 'look' from one of the supported styles.
 - Use a custom trigger
There will be no explicit buy button used to initiate the display of the product. In this case, if the Trigger ID is set, then an "onclick" event will be automatically attached to the page element with the id "trigger_id", otherwise, the merchant needs to arrange for a trigger function to be called when the user clicks on a suitable page element. This is most easily done by adding an onclick function to an element such as an image.
 - Auto-request payment
This option will immediately request the user for payment and is most useful when there is a single component on the page.

4.2. Output of the Wizard

Once complete, the wizard will produce an rmp:protectedContent component ready to be pasted directly into your web page. We recommend that you use the 'inline' option in almost every case, but if the content is very large, it may be better to use a separate file to store it. If you choose NOT to use the 'inline' method, you will also have to first download the encrypted content file to your computer and then upload it to an appropriate folder on your web site. In either case the component uses the "rmp" namespace to avoid any clashes with other components you may be using on the page.

The following snippet is an example output.

```
<rmp:protectedContent
id='pc_img_123'
target_id='showHere'
service_url='http://www.carrot.org/payment'
```

```
product_code='dog_on_a_bicycle'  
merchant_id='LBVJ-LWKV-JWCJ-RGSS'  
price='0.01'  
description='A dog on a bicycle'  
return_url='http://www.mysite.com'  
fail_url='http://www.mysite.com/fail.html'  
trigger_id='image_01'  
buy_button='none'  
style='display: none;'  
4mfCLnz7giXiZ8IufPuCJeJnwi58+4I14mfCLnz7giXmcPRPxfzK0KnJQb8ebbHPEOBUbrqR4gYGFC  
2MbS3qXWN+vdqUBoGktLjii9YLbxlRorVd+Vwvew==  
</rmp:protectedContent>
```

NOTE: The standard 'style' attribute has been added to the component so that it is not displayed directly by the browser.

5 Embedding RMP protected content into a Web page

RMP protected content components (such as the one above), should typically be pasted into a <div> element of an HTML web page and the CarrotPay JavaScript library (RMP-PCL), should be included in the head of the page. Once initialised, RMP-PCL 'looks' for any rmp:protectedContent tags, then creates a single invisible payment <form> for each, finally it inserts the created form into the page just after the corresponding component. The payment form is submitted when triggered (see below), and if successful the protected content is decrypted and placed into the page at the location specified by the target_id.

The following items must be added to the page:

5.1 Namespace declaration

As stated earlier, the protectedContent uses the 'rmp' namespace and this must be declared at the top of the HTML page. The following line must be added to the <html> tag:

```
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:rmp="http://www.carrot.org/rmp">
```

5.2 Javascript library

The following JavaScript library must be included at the top of the page:

```
<script type="text/javascript" src="RMP-PCL.js"/>
```

It is best if you copy the standard rmp library and store it in your html_public directory on your site.

5.3 Initialisation

The RMP-PCL must be initialised to enable it to process the rmp:protectedContent tags. This is most simply done by adding an event handler to the body tag like this:

```
<body onload="rmp.initialise()">
```

However, if you are using other JavaScript on the page or another JavaScript library, you may choose to call the rmp.initialise() function in another way. This is fine as long as this function is called before the user tries to view the content.

5.4 Embed the RMP Protected Content Components into the web page

RMP components (which are generated using the wizard), need to be pasted into the body of your web page. There can be any number of `rmp:protectedContent` components in the same page and they may be placed at any convenient location. However, we recommend that they are placed in a `div` element as follows:

```
<div>
  <rmp:protectedContent ...></rmp:protectedContent>
  ...
  <rmp:protectedContent ...></rmp:protectedContent>
</div>
```

5.5 Set the location where the content is to be displayed

When you create an `rmp:protectedContent` component (either programmatically or using the wizard), you need to specify the location where the content is to be displayed. This is done using the `target_id` attribute and this needs to match an `id` attribute of an HTML element on the page. The most common approach is to use a `div`, so assuming the component uses `target_id="showHere"`, then there will need to be an element like this:

```
<div id="showHere"/>
```

You may use a different `target_id` for each `rmp:protectedContent` component or the same one if you would like the content to be displayed in a fixed location on the page.

5.6 Create an element to trigger the payment request

If the payment request is NOT to be automatic, then one or more elements may be set up to trigger the payment. The two most common methods are a buy button and an image element. In either case you need to arrange to call the `rmp.startPayment()` function. This is most easily achieved by assigning the function to an element's `onclick` event. When the optional "trigger_id" attribute is defined, the html element having that `trigger_id` will automatically have a suitable `onclick` event attached. The `startPayment` function requires a single parameter which is the id of the `rmp:protectedContent` component to be viewed. A typical example is like this:

```

```

6. Generating protected content programmatically

When products or product information are stored in a database, it is not convenient to use a wizard as it is when product creation is ad-hoc. In this case a merchant may still take advantage of `rmp:protectedContent` by generating the component from PHP or Java etc. Table 1. provides a list of all the required attributes and describes how they should be set. The final item is the protected content itself. Content is protected by encrypting it using DES and then base64 encoding the result to make it easier to handle in an HTML page. The DES encryption key is formed from the merchant's hash seed, the price and the product code such that the applied encryption is unique for a given merchant, product and price. CarrotPay has made available a suitable PHP (and later Java), function for server side programmers to create their own protected content which may be used in advance to create protected content files or at runtime during the page creation process.

7. Protected does NOT mean unbreakable

Experienced web developers will realise that `tmp:protectedContent` is a useful tool in preventing users from getting a merchant's content for free but it not a perfect method. Merchants should be aware that due to limitations in current web browser, there is not perfect solution that guarantees that a payment will be made every single time content is viewed. However, if properly executed, merchants may expect this solution to defeat all but the most persistent of attackers.

8. Putting it all together

To summarise the above and by way of examples we offer the following:

For a casual merchant

Start with a 'one-time' registration as a CarrotPay Merchant: [Approx. 3 mins]

- [2 mins] Install a CarrotPurse available from www.carrot.org/download.do
- [1 min] Register as a merchant at www.carrot.org/carrotpay.do

For each product you wish to protect: [Approx. 5 mins each]

- [¼ min] Click the information button on the 'Home' screen of the Merchant wallet and then click the 'Start Web Wizard' button. The Merchant information should be pre-filled so immediately click the 'Next' button.
- [2 mins] Enter your product information. Assumes you have prepared the product code and description and you know your web site details.
- [2 mins] Enter your component information. Assumes you have prepared the html to be protected and decided on the method of triggering payment.
- [¾ min] Copy the component text and paste it into your web page. Assumes you have prepared an html page ready for the component.

For a larger merchant

Start with a 'one-time' registration as a CarrotPay Merchant: [Approx. 3 mins]

- [2 mins] Install a CarrotPurse available from www.carrot.org/download.do
- [1 min] Register as a merchant at www.carrot.org/carrotpay.do

Product preparation: [Approx. 2-3 hours]

- [5-10 mins] Click the information button on the 'Home' screen of the Merchant wallet to reveal your Merchant id, hash seed and service URL. Copy these strings into a suitable file or database table so that they are available to the encryption program.
- [5-10 mins] Download and install the encryption function suitable for your development platform (i.e. PHP or Java).
- [30-60 mins] Design an html page to display your product offering, buttons and display area.
- [30-45 mins] Write a function to select product information from your database and pass it together with your Merchant information to the encryption function.
- [30-45 min] Take the output of the encryption function and either store it in your database for later placement on a web page or store it as content files to be included when pages are generated.

Testing and page refinement: [Approx. 1 day]

A complete web page with protected content

```
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:rmp="http://www.carrot.org.com/rmp">
<script type="text/javascript" src="RMP-PCL.js"/>
<body onload="rmp.initialise()">

<div id="showHere"/>
<div style="display: none">
<rmp:protectedContent
id='1983745172'
target_id='showHere'
service_url='http://www.carrot.org/payment'
product_code='img-animals-dog-funny-112'
merchant_id='LBVJ-LWKV-JWCJ-RGSS'
price='0.01'
description='A dog on a bicycle'
return_url='http://www.mysite.com'
fail_url='http://www.mysite.com/fail.html'
trigger_id='image_01'
buy_button='none'>

4mfCLnz7giXiZ8IufPuCJeJnwi58+4I14mfCLnz7giXmcPRPxfzK0KnJQb8ebbHPEOBUbrqR4gYGFC
2MbS3qXWN+vdqUBoGktLjii9YLbxlRorVd+Vwvew==

</rmp:protectedContent>
</div>
</body>
</html>
```