Signing Web Forms with SetWeb Signer
and
Handling Electronic Signatures with PHP

Jonny Karlsson
Arcada Polytechnic
24.11.2005

GENERATING ELECTRONIC SIGNATURES WITH SETWEB SIGNER

SetWeb Signer is a browser plug-in for generating electronic signatures. This plug-in is included in the smartcard software Setec SetWeb (current version is 1.60). SetWeb Signer is actually an ActiveX component which is embedded in a web page. The component is embedded in the HTML/PHP-code using an <OBJECT>-tag. In the <OBJECT> tag, at least the parameters described in Table 1 need to be defined.

Table 1. Object parameters for embedding the SetWeb Signer plug-in

NAME	VALUE
TYPE	Must be set to "text/x-text-to-sign".
WSXAction	URL of a web page where SetWeb Signer sends the result of the signing operation (signed data).
WSXDataToSign	The data to be signed. Can e.g. be a PHP post variable consisting of data written in a HTML text area (<TEXTAREA>).
WSXBase64	Transport coding based on the Base 64 format. Can be set to "YES" or "NO"
WSXFormat	The format of the signature. Exemples: - "PKCS7SIGNED_Attached" - "PKCS7SIGNED_Detached" "Attached" results in a signature containing the raw data. "Detached" results in a signature containing only a mathematical extract of the raw data.
WSXName	The name of the variable containing the signature in URL-coded format. This variable is accessible via the PHP Post-table in the source code of the web-site to which the signature is posted to.

A special submit button, "WSXButton" is needed to enable the user to activate the SetWeb Signer signing window. This button is embedded using with an <EMBED> tag within the <OBJECT> tag. See the code example in figure 1.

```
<OBJECT CLASSID="CLSID:8AC8A833-2F0F-11D5-845D-0050DA2DEE56" WIDTH="120" HEIGHT="35">
    <PARAM NAME="TYPE" VALUE="text/x-text-to-sign">
    <PARAM NAME="WSXAction" VALUE="https://hst2.arcada.fi/~testila/test2/verifiera_signatur.html">
    <PARAM NAME="WSXDataToSign" VALUE=" <?php echo $_POST["varablename_of_textarea"]; ?>">
    <PARAM NAME="WSXBase64" VALUE="YES">
    <PARAM NAME="WSXFormat" VALUE="PKCS7SIGNED_Attached">
    <PARAM NAME="WSXName" VALUE="signedData">

    <embed
        WIDTH="120" HEIGHT="35"
        WSXButtonName="Sign"
    </embed>
</OBJECT>
```

Figure 1. Embedding SetWeb Signer in the HTML code.

HANDLING AND VERIFYING ELECTRONIC SIGNATURES WITH PHP

The signatures generated with SetWeb Signer can be verified using the function *openssl_pkcs7_verify()*. This function is a standard function in the PHP API in case PHP was compiled with OpenSSL on the web server. The openssl_pkcs7_verify() function requires that the signature information is in clear text and in S/MIME-format. Therefore, the signature information generated by SetWeb Signer must be modified before the PHP-function is able to verify the signature. The S/MIME format is reached by splitting the signature in rows consisting of 64 characters each and adding the S/MIME header, see figure 2.

```
$signature = $_POST[“the_value_of_WSXName”];

$signature = chunk_split($signature, 64);
$signature = "MIME-Version: 1.0\nContent-Disposition: attachment;
filename=\"smime.p7m\"\nContent-Type: application/x-pkcs7-mime;
name=\"smime.p7m\"\nContent-Transfer-Encoding: base64\n\n".$signature;
```

Figure 2. Converting the signature information generated by SetWeb Signer to S/MIME format.

The function *openssl_pkcs7_verify()* is called by giving four parameters:

Param1 - Path to the file including the signature.
Param2 - A flag defining the type of the signature.
Param3 - Path to the file where the signature certificate will be stored.
Param4 - An array containing the paths to all trusted CA certificate files.

The verifying function requires that the signature to be verified is stored in a file. Therefore, the signature information must be stored in a temporal file before calling the function. If the signature is verified successfully openssl_pkcs7_verify() returns “true”, otherwise “false”. An example of how a S/MIME signature can be verified is shown in figure 3.

```
$ca = array();
$ca[0] = “path to ca-file1”;
$ca[1] = “path to ca-file2”;
…

$sign_cert_file = “sign_cert_file”;
$tmpfile = “temp_sig_file”;
$handler = fopen($tmpfile, “w”);
fwrite($handler, “$signature”);
fclose($handler);

$status = openssl_pkcs7_verify($tmpfile, PKCS7_BINARY, $sign_cert_file, $ca);
```

Figure 3. Verifying a S/MIME signature in PHP

Identity information of the signer certificate can be examined for example by extracting the subject field from the signer certificate. This can be done as shown in Figure 4. by using the PHP functions *file_get_contents()* and *openssl-x509_parse()*.

```
$cert = file_get_contents($sign_cert_file);
$ssl = openssl_x509_parse($cert);
$signerinfo = $ssl['name'];
```

Figure 4. Extracting the subject field from the signer certificate.