KryptoKoder

Personal Cryptographic Tools

Signing a PDF document with PHP, Ruby or Python

MyPDFSigner provides extensions for PHP, Ruby and Python to sign PDF documents using tokens stored in PKCS#12 certificate stores. The extensions support timestamping (RFC 3161) and visible signatures, including the incorporation of "watermark" images. See example below.

Quick Start Guide

Install desktop application:
-- rpm -ihv mypdfsigner-1.2.4-1.i686.rpm (Fedora)
-- dpkg -i mypdfsigner_1.2.4-1_i386.deb (Ubuntu)
Test command line:
-- mypdfsigner -i /usr/local/mypdfsigner/tests/example.pdf -o /tmp/example-signed.pdf -z /usr/local/mypdfsigner/tests/mypdfsigner.conf -v -q
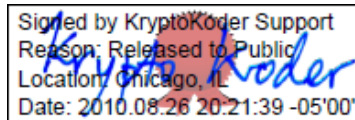Check /tmp/example-signed.pdf; Install extension:
-- rpm -ihv mypdfsigner-[php|ruby|python]-0.8.0-1.i686.rpm (Fedora)
-- dpkg -i mypdfsigner-[php|ruby|python]_0.8.0-1_i386.deb (Ubuntu)
Test a script from the command line:
-- [php|ruby|python] /usr/local/mypdfsigner/tests/test.[php|rb|py]
Check /tmp/example-signed-[php|ruby|python].pdf

Note: MyPDFSigner for Linux provides two command line tools that are very similar, mypdfsigner-cli and mypdfsigner. The first uses iText Java code to do the signing (the same happens with the graphical application) while the second uses KryptoKoder's C code that is also used by the extensions.

Configuration

Before using any of the extensions it is convenient to start with the desktop application to create a configuration file for the key store and alias one wants to use. The desktop application creates a .mypdfsigner file in your home directory. That file can be copied to /usr/local/mypdfsigner and renamed mypdfsigner.conf (this step can be skipped but then the configuration file needs to be specified when calling the "sign" function).

MyPDFSigner supports visible signatures (on the first page) and allows for some basic customization (image, size and position). A signature is made visible by setting TRUE the "visible" argument of the "sign" function (or by passing -v when using the command line).

Before explaining how signature customization is done one needs to know about PDF size units. Units are based on a "72 units per inch" scale. Hence letter size (8.5 by 11 inches) corresponds to 612 x 792 units, and A4 (210 by 297 mm) corresponds to 595 x 842 units. A visible signature position is specified by an array of four values corresponding to the llx (lower left x), lly (lower left y), urx (upper right x) and ury (upper right y) coordinates of the lower left and upper right corners of the signature rectangle. By default MyPDFSigner uses the rectangle [442 712 572 752]. A different signature rectangle can be specified in the configuration file by adding the entry sigrect=[llx lly urx ury] with the new values. The text part of the visible signature consists of four lines: the cn (common name, obtained from the certificate), reason, location and date. The font size is adapted to the specified rectangle height. However the width of the rectangle needs to be manually tuned so that the text fits.

A visible signature can also incorporate an image if the "sigimage" entry is present in the configuration file. The image will be scaled to fit inside the signature rectangle. The suggested approach to select the right sized signature image is to start by choosing the right visible signature rectangle size (as explained above). Once that is known create a signature image with the same proportions (but high enough resolution so that it looks fine when printed). An example: if the rectangle is 130 (units) wide by 40 tall (like the default one) the size when printed will be 130/72 inches by 40/72 inches. Sign your name in a piece of paper inside a rectangle of such dimensions and scan it at, say, 300 dpi. This will create an image that is

Signed by KryptoKoder Support
Reason: Released to Public
Location: Chicago, IL
Date: 2010.08.26 20:21:39 -05'00'

Timeline

2011-04-27
Released MyPDFSigner 1.2.1 with support for OpenSC and PKCS#11 token based batch signing. **Permite assinatura em massa com o Cartão de Cidadão.**

2011-01-13
Released MyDWFSigner 0.8.0, a tool to sign DWF documents.

2010-09-24
Released MyPDFSigner 1.1.6 with support for configurable visible signatures.

2010-06-16
Released MyPDFSigner 1.1.0 with support for Time Stamping.

2010-04-14
Released MyPDFSigner 1.0.5 for Mac OS X only with support for the Apple Keychain Store.

2010-03-19
Released MyPDFSigner 1.0.0 with support for PKCS#12 files and a command line interface.

2009-12-05
Released MyPDFSigner 0.9.5 for Windows only with support for the Windows Certificate Store.

2009-11-10
Released MyPDFSigner 0.9.0 with support for all PKCS11 cards.

2009-10-19
Released MyPDFSigner 0.8.0 with support for the Portuguese Citizen Card only. **Funciona com o Cartão de Cidadão.**

130*300/72 (pixels) wide by 40*300/72 tall (or 542 by 167 pixels). An image with such resolution will scale nicely and the resulting graphics will have the right size. Images of different proportions can be used but since they will be scaled (and centered) to fit inside the rectangle there will be space around two of the sides of the image.

MyPDFSigner has some limitations regarding the type of images that can use (another limitation is the fact that it does not yet support "updated" PDF documents). The image needs to be of RGB-Alpha PNG type. That is not a serious limitation since it is the default format used by Gimp when saving a color PNG file that includes a transparent layer. In case of doubt look at the image properties using Gimp.

An example of a signed and timestamped PDF document with a visible signature is available here. Note that it was signed with a self signed certificate so the warning one sees when opening in Adobe Reader is expected.

MyPDFSigner API

MyPDFSigner does just one thing: it signs PDF documents. As such it provides just one function. Examples of its usage are shown next.

PHP Example

```php
<?php

$originalPath = "/tmp/test.pdf";
$signedPath = "/tmp/test-signed.pdf";
$location = "Chicago";
$reason = "Testing";
$contactInfo = "support@kryptokoder.com";
$certify = FALSE;        // not supported yet; defaults to FALSE
$visible = TRUE;
$author = "KryptoKoder";
$tile = "Signing with MyPDFSigner";
$subject = "PHP Extension";
$keyowrds = "KryptoKoder, PKCS#12, PDF";
$confFile = ""; // defaults to /usr/local/mypdfsigner/mypdfsigner.conf if empty
$timestamp = TRUE;

echo mypdfsigner_sign($originalPath, $signedPath, $location, $reason, $contactInfo,
    $certify, $visible, $author, $title, $subject, $keywords, $confFile, $timestamp);

?>
```

Ruby Example

```ruby
require 'mypdfsigner'
include MyPDFSigner

inputPath = "/tmp/input.pdf"
outputPath = "/tmp/output.pdf"
location = "Chicago"
reason = "Demo"
contactInfo = "+1 555-555-5555"
certify = false   # not supported yet
visible = true
title = "Signing with MyPDFSigner"
author = "KryptoKoder"
subject = "Ruby Extension"
keywords = "PKCS#12, MyPDFSigner, PDF"
confFile = ""   # defaults to /usr/local/mypdfsigner/mypdfsigner.conf if empty
timestamp = true

puts mypdfsigner_sign(inputPath, outputPath, location, reason, contactInfo,
    certify, visible, title, author, subject, keywords, confFile, timestamp)
```

Python Example

```python
import mypdfsigner
```

```
inputPath = "/tmp/input.pdf"
outputPath = "/tmp/output.pdf"
location = "Chicago, Illinois"
reason = "Demo"
contactInfo = "+1 555-555-5555"
certify = False     # not supported yet; defaults to false
visible = True
title = "Signing with MyPDFSigner"
author = "KryptoKoder"
subject = "Python Extension"
keywords = "PKCS#12, PDF, MyPDFSigner"
confFile = ""   # defaults to /usr/local/mypdfsigner/mypdfsigner.conf if empty
timestamp = True

print mypdfsigner.sign(inputPath, outputPath, location, reason, contactInfo,
    certify, visible, title, author, subject, keywords, confFile, timestamp)
```