PDF Encryption

pdfencrypt info

Lombard Business Park
8 Lombard Road
Wimbledon
London, ENGLAND SW19 3TZ

103 Bayard Street
New Brunswick
New Jersey, 08901
USA

Pdfencrypt Info

Contents

Pdfencrypt Info

August 2002

This document provides a basic introduction to using PdfEncrypt. It includes explanations of what PdfEncrypt does, how it can be used, what the current limitations are, how to run PdfEncrypt as a command line program, how to use PdfEncrypt with the ReportLab RML2PDF and PageCatcher applications, and how to use the Pdfencrypt programming interface within other programs.

What is PdfEncrypt for?

PdfEncrypt is an add-on utility for ReportLab's suite of enterprise reporting tools. It allows you to encrypt PDF files, add user and owner passwords and allow or prevent users access to certain capabilities such as printing the file or copying and pasting from the file.

How can this be useful in the real world? Imagine a situation where you are generating customised reports with sensitive customer-specific information (for example a bank statement). PdfEncrypt makes it possible for your user to get a PDF file with the same password that they use to login to your web site which is safe and secure to send out by email.

Consider revenue protection. Imagine you are an organisation which sells market research reports for $500 each. Your customers will find it a lot less easy to pass your reports around if the front page of each report is customised to say "prepared exclusively for Fred Bloggs, no one else should see this" and it's password protected with a password like "fbloggs:v1J99x"

For maximum effect, combine PdfEncrypt with our PageCatcher product, and either our Open Source Library or RML2PDF to personalize as well as protect.

About encrypting PDF files

Adobe's PDF standard allows you to do three related things to a PDF file when you encrypt it:

- Apply password protection to it, so a user must supply a valid password before being able to read it,
- Encrypt the contents of the file to make it useless until it is decrypted, and
- Control whether the user can print, copy and paste or modify the document while viewing it.

The PDF security handler allows two different passwords to be specified for a document:

- The 'owner' password (*aka* the 'security password' or 'master password')
- The 'user' password (*aka* the 'open password')

When a user supplies either one of these passwords, the PDF file will be opened, decrypted and displayed on screen. If the owner password is supplied, then the file is opened with full control - you can do anything to it, including changing the security settings and passwords, or re-encrypting it with a new password. If the user password was the one that was supplied, you open it up in a more restricted mode. The restrictions were put in place when the file was encrypted, and will either allow or deny the user permission to do the following:

- Modifying the document's contents,
- Copying text and graphics from the document,
- Adding or modifying text annotations and interactive form fields,
- Printing the document.

Note that all password protected PDF files are encrypted, but not all encrypted PDFs are password protected. If a document's user password is an empty string, there will be no prompt for the password when the file is opened. If you only secure a document with the owner password, there will also not be a prompt for the password when you open the file. If the owner and user passwords are set to the same string when encrypting the PDF file, the document will always open with the user access privileges. This means that it is possible to create a file which, for example, is impossible for anyone to print out, even the person who created it.

Owner password set?	User password set?	Result
Y	-	No password required when opening file. Restrictions apply to everyone.
-	Y	User password required when opening the file. Restrictions apply to everyone.
Y	Y	A password required when opening the file. Restrictions apply only if user password supplied.

When a PDF file is encrypted, encryption is applied to all the strings and streams in the file. This prevents people who don't have the password from simply removing the password from the PDF file to gain access to it - it renders the file useless unless you actually have the password. PDF's standard encryption methods use the MD5 message digest algorithm (as described in RFC 1321, *The MD5 Message-Digest Algorithm*) and an encryption algorithm known as RC4. RC4 is a symmetric stream cipher - the same algorithm is used both for encryption and decryption, and the algorithm does not change the length of the data.

If an encrypted PDF file is opened using Adobe Acrobat (but not Acrobat Reader), a small yellow key appears in the bottom status bar.

How do I use PdfEncrypt?

Command line usage

From a Windows command prompt ('DOS box'), type the following

```
pdfencrypt.exe -h
```

From a Unix shell, type the following

```
python pdfencrypt.pyc -h
```

This produces a usage reminder which should look like the one below:

```
C:\Python\rlextra\utils>pdfencrypt.exe -h
PDFENCRYPT USAGE:

PdfEncrypt encrypts your PDF files.

Line mode usage:

% pdfencrypt.exe pdffile [-o ownerpassword] | [owner ownerpassword],
        [-u userpassword] | [user userpassword],
        [-p 1|0] | [printable 1|0],
        [-m 1|0] | [modifiable 1|0],
        [-c 1|0] | [copypastable 1|0],
        [-a 1|0] | [annotatable 1|0],
        [-s savefilename] | [savefile savefilename],
        [-v 1|0] | [verbose 1|0],
        [-e128], [encrypt128],
        [-h] | [help]

-o or owner set the owner password.
-u or user set the user password.
-p or printable set the printable attribute (must be 1 or 0).
-m or modifiable sets the modifiable attribute (must be 1 or 0).
-c or copypastable sets the copypastable attribute (must be 1 or 0).
-a or annotatable sets the annotatable attribute (must be 1 or 0).
-s or savefile sets the name for the output PDF file
-v or verbose prints useful output to the screen.
     (this defaults to 'pdffile_encrypted.pdf').
'-e128' or 'encrypt128' allows you to use 128 bit encryption (in beta).

-h or help prints this message.

See PdfEncryptIntro.pdf for more information.


C:\Python\rlextra\utils>
```

This should tell you most of what you need to know about the command line options for PdfEncrypt, but here are some more specific notes on them

- The first argument after pdfencrypt should always be the filename of the input file (referred to as `pdffile` above). This should be an unencrypted PDF file.
- All arguments other than the input filename have both a long version (a memorable word) and a short version (beginning with a minus sign, and shorter to type). Both versions will act in exactly the same manner.
- All arguments other than the input filename are optional (but you must use at least one of them - there's no point giving pdfencrypt a filename but not telling it what to do with it!). Where arguments are given, they should be separated by a space (eg '`-s outfile.pdf`' rather than '`-soutfile.pdf`' or '`-s:outfile.pdf`').
- The output file should be specified using the '`-s`' or '`savefile`' argument. If no savefile argument is given, the filename for the output file defaults to the input filename with the suffix of

'_encrypted.pdf' added.

- If the '-o' or 'owner' argument is given, the word following this becomes the owner password (see the section headed 'About encrypting PDF files' for more information about owner passwords).
- If the '-u' or 'user' argument is given, the word following this becomes the user password (see the section headed 'About encrypting PDF files' for more info).
- The remaining arguments act as flags, switching on or off an ability when the user views the encrypted PDF. For all of these, the argument must be followed by a '0' (to switch it off), or a '1' to switch it on. The default for all of these is 1 (enabled). The arguments affected by this are:

  '-a', 'annotatable',
  (These turn on or off the user's ability to add new or modify existing text annotations and interactive form fields),

  '-c', 'copypastable',
  (These turn on or off the user's ability to copy and paste text and/or images from the document),

  '-m', 'modifiable',
  (These turn on or off the user's ability to modify the documents contents),

  '-p', 'printable'
  (These turn on or off the user's ability to print out the document).

Examples

The following example encrypts the file 'test.pdf', giving it an owner password of 'SPAM!' and a user password of "SPAM". The -v flag sets the 'verbose' attribute so output is echoed onto the screen (giving us some feedback). We are setting the Modifiable and Copypastable attributes to zero, but still allowing the user who opens the output PDF to annotate it and print it out. Because we haven't given an output filename, the output PDF defaults to 'test_encrypted.pdf' ('test' plus the suffix of '_encrypted.pdf').

```
C:\Python\rlextra\utils>pdfencrypt.exe test.pdf -v 1 -o SPAM! -u SPAM -m 0 -c 0
Owner password set to: 'SPAM!'.
User password set to: 'SPAM'.
'Modifiable' set to: '0'.
'Copypastable' set to: '0'.
wrote output file 'test_encrypted.pdf'(42410 bytes)
  owner password is 'SPAM!'
  user password is 'SPAM!'

C:\Python\rlextra\utils>
```

The following example also encrypts the file 'test.pdf', but uses the longer versions of the arguments. The user password is set to "Eggs". The owner password is set to "I DON'T LIKE SPAM!" - notice how it has to be included in quotation marks since it contains spaces. Printable is set to 0 - the other document attributes will stay at the default 1, allowing the user to annotate, copy and paste and modify the document. Here we do specify the output filename - 'nospam.pdf'.

```
C:\Python\rlextra\utils>pdfencrypt.exe test.pdf user Eggs owner "I DON'T LIKE SPAM!"
verbose 1 printable 0 savefile nospam.pdf
Owner password set to: 'I DON'T LIKE SPAM!'.
User password set to: 'Eggs'.
'Printable' set to: '0'.
Output file set to: 'nospam.pdf'.
wrote output file 'nospam.pdf'(42456 bytes)
  owner password is 'I DON'T LIKE SPAM!'
  user password is 'Eggs'

C:\Python\rlextra\utils>
```

Another example. This one is identical to the previous one, only using the short form of the arguments. It also differs in that the verbose flag is set to 0, so no output is produced. This can be useful in situations where

printing to the standard output is not desirable.

```
C:\Python\rlextra\utils>pdfencrypt.exe test.pdf -u Eggs -o "I DON'T LIKE SPAM!" -v 0
-p 0 -s nospam.pdf

C:\Python\rlextra\utils>dir *.pdf
 Volume in drive C has no label.
 Volume Serial Number is 07D1-020F

 Directory of C:\Python\rlextra\utils

20/08/2002  17:24              42,409 nospam.pdf
               1 File(s)         42,409 bytes
               0 Dir(s)   3,125,477,376 bytes free

C:\Python\rlextra\utils>
```

One last example. When you do not have a licensed copy of PdfEncrypt, the passwords default to hardcoded ones:

```
C:\Python\rlextra\utils>pdfencrypt.exe test.pdf -u Eggs -o "I DON'T LIKE SPAM!" -v
1 -p 0 -s nospam.pdf
Owner password set to: 'This is Unlicensed Software' (evaluation version).
User password set to: 'Mr Unlicensed Software' (evaluation version).
'Printable' set to: '1' (evaluation version).
Output file set to: 'encrypted.pdf' (evaluation version).
wrote output file 'test_encrypted.pdf'(42434 bytes)
  owner password is 'This is Unlicensed Software'
  user password is 'Mr Unlicensed Software'

C:\Python\rlextra\utils>
```

Calling from Python

From Python, you need to import the pdfencrypt module before you can do anything with it.

```
import rlextra.utils.pdfencrypt
```

Once this is done, you can use the functions *encryptPdfOnDisk* and *encryptPdfInMemory*:

encryptPdfOnDisk

This creates the output PDF on disk. The argument footprint looks like this:

```
encryptPdfOnDisk(inputFileName, outputFileName, userPassword,
                 ownerPassword=None, canPrint=1, canModify=1, canCopy=1, canAnnotate=1)
```

inputFileName	The name of the input PDF file (required).
outputFileName	The name to be used for the output (encrypted) PDF file (required).
userPassword	The user password to be used (required)
ownerPassword	The owner password to be used (optional).
canPrint	Can the user print the output PDF? (defaults to 1)
canModify	Can the user modify the contents of output PDF? (defaults to 1)
canCopy	Can the user copy and paste text/images from the output PDF? (defaults to 1)
canAnnotate	Can the user make annotations on the output PDF? (defaults to 1)

returns: the length of the output PDF file in bytes.

examples:

```
C:\Python>python
Python 2.2.1 (#34, Apr  9 2002, 19:34:33) [MSC 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import rlextra.utils.pdfencrypt
>>> rlextra.utils.pdfencrypt.encryptPdfOnDisk('test.pdf', 'saved.pdf', 'SPAM',
canPrint=0)
42307

>>> from rlextra.utils.pdfencrypt import encryptPdfOnDisk
>>> encryptPdfOnDisk('test.pdf', 'saved2.pdf', 'SPAM', ownerPassword="SPAM!",
canPrint=0, canCopy=0)
42346
>>>
```

encryptPdfInMemory

This function accepts a PDF file 'as a byte array in memory', encrypts it and returns encrypted one.

This is a high level convenience and does not touch the hard disk in any way. If you are encrypting the same file over and over again, it's better to use PageCatcher and cache the results (see the later section on how to use PdfEncrypt and PageCatcher).

The argument footprint looks like this:

```
encryptPdfInMemory(inputPDF, userPassword, ownerPassword=None,
                   canPrint=1, canModify=1, canCopy=1, canAnnotate=1)
```

inputFileName	The name of the input byte array (required).
userPassword	The user password to be used (required)

ownerPassword	The owner password to be used (optional).
canPrint	Can the user print the output PDF? (defaults to 1)
canModify	Can the user modify the contents of output PDF? (defaults to 1)
canCopy	Can the user copy and paste text/images from the output PDF? (defaults to 1)
canAnnotate	Can the user make annotations on the output PDF? (defaults to 1)

returns: an encrypted version of the input array.

example:

```
C:\Python>python
Python 2.2.1 (#34, Apr  9 2002, 19:34:33) [MSC 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import rlextra.utils.pdfencrypt
>>> input = open('test.pdf', 'rb').read()
>>> output = rlextra.utils.pdfencrypt.encryptPdfInMemory(input, 'SPAM', canPrint=0)
>>>

>>> from rlextra.utils.pdfencrypt import encryptPdfInMemory
>>> output2 = encryptPdfInMemory(input, 'SPAM', ownerPassword="SPAM!", canPrint=0,
canCopy=0)
>>>
```

Use from the ReportLab Toolkit

The ReportLab toolkit is our free, Open Source library for creating PDFs from Python. You can use the function *encryptCanvas* and the class *EncryptionFlowable* with it.

encryptCanvas

The encryptCanvas applies encryption to the document being generated. You must pass in a reportlab canvas object, which it then encrypts.

The argument footprint looks like this:

```
encryptCanvas(canvas, userPassword, ownerPassword=None,
                  canPrint=1, canModify=1, canCopy=1, canAnnotate=1)
```

canvas	The canvas to be encrypted (required).
userPassword	The user password to be used (required)
ownerPassword	The owner password to be used (optional).
canPrint	Can the user print the output PDF? (defaults to 1)
canModify	Can the user modify the contents of output PDF? (defaults to 1)
canCopy	Can the user copy and paste text/images from the output PDF? (defaults to 1)
canAnnotate	Can the user make annotations on the output PDF? (defaults to 1)

returns: None.

example:

```
C:\Python\rlextra\utils>python
Python 2.2.1 (#34, Apr  9 2002, 19:34:33) [MSC 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from reportlab.pdfgen import canvas
>>> c = canvas.Canvas("hello.pdf")
>>> from rlextra.utils.pdfencrypt import encryptCanvas
```

```
>>> encryptCanvas(c, "SPAM")
>>> c.drawCentredString(297, 420, "Hello World")
>>> c.showPage()
>>> c.save()
```

EncryptionFlowable

This class is a flowable, and can be used just like an other flowables. It can be dropped into your Platypus story and it will set up the encryption options. Like other flowables, it has to be drawn onto the canvas (see below for an example of how to do this).

If you have multiple instances of EncryptionFlowable in a document, then the encryption setting that are used come from the last one to be called.

See Chapter 5 of the ReportLab User Guide for more information on using Platypus)

The argument footprint for starting an new instance of EncryptionFlowable looks like this:

```
EncryptionFlowable(canvas, userPassword, ownerPassword=None,
                   canPrint=1, canModify=1, canCopy=1, canAnnotate=1)
```

canvas	The canvas to be encrypted (required).
userPassword	The user password to be used (required)
ownerPassword	The owner password to be used (optional).
canPrint	Can the user print the output PDF? (defaults to 1)
canModify	Can the user modify the contents of output PDF? (defaults to 1)
canCopy	Can the user copy and paste text/images from the output PDF? (defaults to 1)
canAnnotate	Can the user make annotations on the output PDF? (defaults to 1)

```
from reportlab.pdfgen import canvas
from reportlab.lib.styles import getSampleStyleSheet
from reportlab.platypus import Paragraph
from rlextra.utils.pdfencrypt import EncryptionFlowable
styleSheet = getSampleStyleSheet()
style = styleSheet['BodyText']
P1=Paragraph('This is an encrypted document. ',style)
aW = 460
aH = 800
c = canvas.Canvas("hello.pdf")
ef = EncryptionFlowable('SPAM', ownerPassword="SPAM!")
w,h = P1.wrap(aW, aH)
P1.drawOn(c,20,aH)
ef.drawOn(c,20,aH)
c.save()
```

Calling from other commercial ReportLab products (RML2PDF and PageCatcher)

PageCatcher

```
To do
```

RML2PDF

```
To do
```

Known Deficiencies and Caveats

If the owner password is not set, and the user password *is* set, then PdfEncrypt will set both passwords to the user password. Since they will both the same, the file will always open with the user access privileges. This means that it is possible to create encrypted files for which nobody has permission to modify, print etc.

Demo Modes

If you have not purchased a license for PdfEncrypt, you are still able to use it. However, all the user restrictions will be set to 1 (so for example, you won't be able to prevent a user printing out a document). Also, if you set the user password it will be reset to a hardcoded one (currently 'Mr Unlicensed Software'), and if you set an owner password, it will be set to the hardcoded 'This is Unlicensed Software'.

All these restrictions are removed if you purchase a license. Email info@reportlab.com for more details.

Feedback

We need and welcome feedback to help make this into a great product! Email info@reportlab.com, or join our group of 200+ existing users by emailing reportlab-users@reportlab.com (with 'Subscribe' in the subject line), or join from the web interface at http://two.pairlist.net/mailman/listinfo/reportlab-users.

Enjoy!

Appendix: about Acrobat Encryption

To do