PDF Encryption
Research seminar in cryptography

Yauhen Yakimenka

December 15, 2015

1 Introduction

PDF files are arguably the most popular format for electronic documents. In this report we talk about security of password protection of PDF files, what kind of encryption is used and in which way.

The contents of the report are as follows. Section 2 describes the inner format of PDF files. And section 3 concentrates on PDF password protection itself.

Technical details of the report are based on [1].

2 PDF file structure

Strictly speaking, PDF file is a binary file, i.e. a sequence of bytes. However, all the controlling elements of the file use only lower 7 bits of the byte and are plain-text if seen in ASCII encoding.

Moreover, the whole file could be represented as a sequence of ASCII characters through usage of escape sequences. However, literal strings and stream objects may use all 8 bits of the byte and in most practical applications they do. No particular encoding is imposed by PDF standard on the bytes with values 128–255; they should be considered on a base level just as byte values. See 2.2 for more details).

2.1 High-level PDF file structure

Basic high-level structure is shown at the Figure 2.1.

Header	`%PDF-1.5`
	`%âãÏÓ`
Body	`1 0 obj`
	`...`
Cross-reference table	`xref`
	`0 16`
	`0000000000 65535 f`
	`...`
Trailer	`trailer`
	`<<`
	`/Info 15 0 R`
	`>>`
	`startxref`
	`10487`
	`%%EOF`

Figure 1: High-level structure of PDF file

Some notes on the file structure:

- Lines starting with % are considered as comments and ignored by PDF processing applications.
- The first line is the version of PDF file. File version `1.n` corresponds to Adobe Acrobat software version `n.xx` (where `xx` is a subversion of software).
- It is conventional to put in the second line of the file comment with some non-ASCII characters so that file type analysing software (like it is common in Linux systems) know that the file is not a text file.
- Body is a sequence of PDF objects. They represent all the contents of the PDF file.
- Cross-reference table is a "table of contents" of PDF file, which holds the list of all objects in the file as well as offset in the file (in bytes) where the object begins. It simplifies the navigation in random-access mode.

- Trailer contains general information about the file, for example the number of the object that holds information about encryption of the file.
- After `startxref` there is an integer number which is a position in the file (an offset in bytes from the beginning of the file) where cross-reference table starts.

The information about the file encryption is kept in an indirect dictionary-type object referred in the trailer. Example of such an object:

```
14 0 obj
<<
/V 2
/Filter /Standard
/U (...)
/Length 128
/R 3
/P -3904
/O (...)
>>
endobj
```

We omitted contents of literal strings (inside of "(...)") because they contain non-printing characters.

2.2 Object file types

Here we briefly talk about the types of objects in PDF file. Because for purposes of this report we do not need boolean objects and stream objects, we omit their description.

Numeric Numeric objects are integers and reals in fixed-dot format.

Literal stings This is the first way to represent byte strings in PDF. In this case the string is written "as is" and delimited by `(` and `)`, although escape sequences (like `\n`, `\t`, `\\`), etc.) can be used:

```
(This is a literal string,\nthat contains two lines)
```

Hexadecimal strings This is another way to represent the string. A hexadecimal string is written as a sequence of hexadecimal digits (0–9 and either A–F or a–f) enclosed within angle brackets (< and >):

```
<4E6F762073686D6F7A206B6120706F702E>
```

Name objects These are just names, which can contain any symbols except whitespace symbols:

```
/Name1
/@@!another--
/
/Previous-name-was-empty-name
```

Array objects Array object is a sequence of heterogeneous objects. In particular, it can contain nested arrays.

```
[12  (Hello, world!\n)  -.03  [1 2 /Encode] <34FE2A>]
```

Dictionary objects Dictionary is an associative array. Keys are always name objects while values could be of any type (including dictionary).

```
<< /Type /Example
   /Subtype /DictionaryExample
   /Version 0.01
   /IntegerItem 12
   /StringItem ( a string )
   /Subdictionary << /Item1 0.4
                     /Item2 true
                     /LastItem ( not ! )
                     /VeryLastItem ( OK )
                  >>
>>
```

Indirect objects Any object in PDF file can be labelled as *indirect*. That gives an object a unique object identifier by which other objects can refer it. Indirect object is written as object number and generation number, followed by the value of the object bracketed between keywords `obj` and `endobj`, e.g.

```
14 0 obj
    [(This is indirect array) 134 .001 [<0fde><24de>]]
endobj
```

This object can be then referred as `14 0 R` wherever the original object should be placed, e.g.

```
<< /Version 1
   /Subversion .1
   /Data 14 0 R
>>
```

Stream objects To put it simply, stream objects are byte strings of unlimited length. We do not discuss their representation as we do not need it for the project.

3 PDF protection

3.1 Owner password and user password

PDF's standard security handler allows access permissions and up to two passwords to be specified for a document: an *owner password* and a *user password*.

If a user attempts to open an encrypted document that has a user password, the application should prompt for a password. Correctly supplying either password enables the user to open the document, decrypt it, and display it on the screen. If the document does not have a user password, no password is requested; the application can simply open, decrypt, and display the document. Whether additional operations are allowed on a decrypted

document depends on which password (if any) was supplied when the document was opened and on any access restrictions that were specified when the document was created:

- Opening the document with the correct owner password (assuming it is not the same as the user password) allows full (owner) access to the document. This unlimited access includes the ability to change the document's passwords and access permissions.
- Opening the document with the correct user password (or opening a document that does not have a user password) allows additional operations to be performed according to the user access permissions specified in the document's encryption dictionary.

User access permissions are like modifying the document's contents, copying, printing, etc. However, it is very important to understand that if the document has been successfully opened and decrypted, the application has access to the entire contents of the document. So it is up to the application to control the user rights to perform certain operations.

Further in this report we will discuss proper encryption only but not the other permissions restrictions since they are obviously insecure. It is only string and stream objects that are encrypted. This is so because they are the object containing the document contents. All the other objects describe metadata. This is also useful if we need only random access to the objects (for example, viewing only one page at a time) as it allows to decrypt only those parts of the document we need. Full encryption would disallow to do this.

3.2 PDF encryption primitives

The following cryptographic basic algorithms are used in PDF:

RC4 is a symmetric stream cipher. The length of the data does not changed.

AES is a symmetric block cipher. The length of the data is rounded up to to a multiple of the block size. In PDF block size is always 16 bytes. The padding scheme used with AES is PKCS#5.

MD5 is a message-digest algorithm.

In the next subsection we will discuss how these standard algorithms are used to encrypt data.

3.3 PDF encryption algorithms

The following values are used as input for encryption routines:

password(s)	User and, probably, owner passwords. User password is required for encryption. Owner password is used only if it is set;
pad str	Fixed 32-bytes-long string used for padding.
/Length	Length of the key. Value is stored in encryption dictionary object in PDF file.
/ID	It is array of two byte strings that are generated when PDF file is first time created. There are no strict rules on how to generate it. Stored in trailer.
/P	4-byte mask that represents allowed actions (not secure as it depends on PDF viewer software only). It is stored in file encryption dictionary object as signed integer.

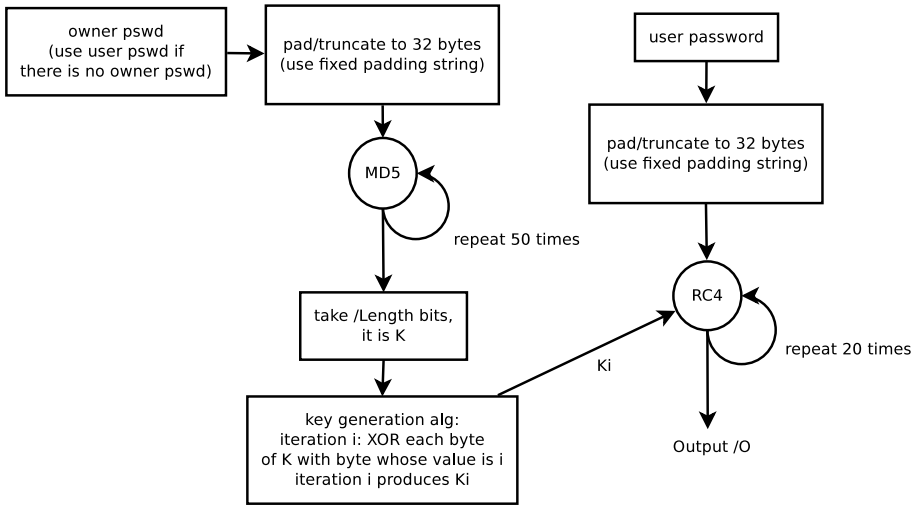

Figure 2: Algorithm to produce `/O`.

With these values and algorithm at Figure 2 one can produce `/O` value, that is stored in encryption dictionary object.

/O value is then used in algorithm at Figure 3 to produce an encryption key.

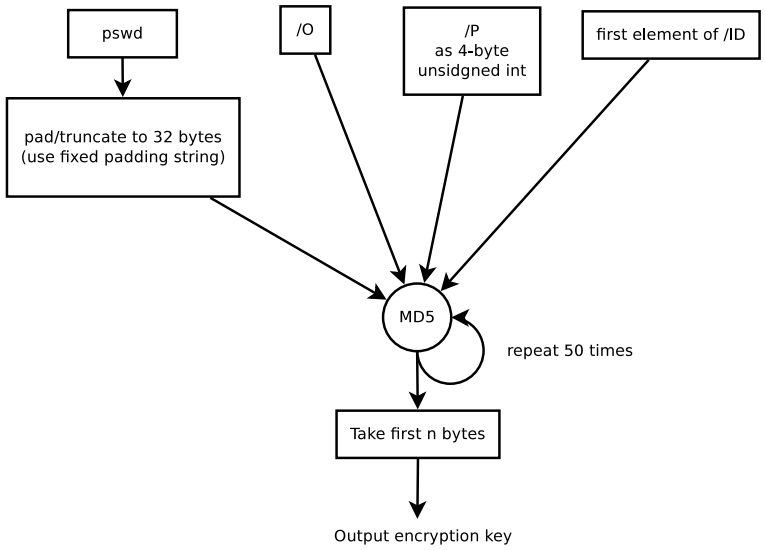

Figure 3: Algorithm to produce the encryption key n bytes long.

Further, with algorithm at Figure 4 one can produce /U value, that is also stored in encryption dictionary object.

Algorithm to check the validity of the password is easy. One runs algorithm from Figure 4 except the last step and compares obtained value with first 16 bytes of /U value stored in PDF file.

3.4 Proof-of-concept script

The proof-of-concept script has been developed in Python 3. The repository of the script is available here:
https://bitbucket.org/jjauhien/pdfcrypt/src.

3.5 Security of PDF encryption

As it was mentioned before, restriction on allowed actions is not secure at all, as it is controlled entirely on reader software.

Having empty owner password breaks all the encryption.

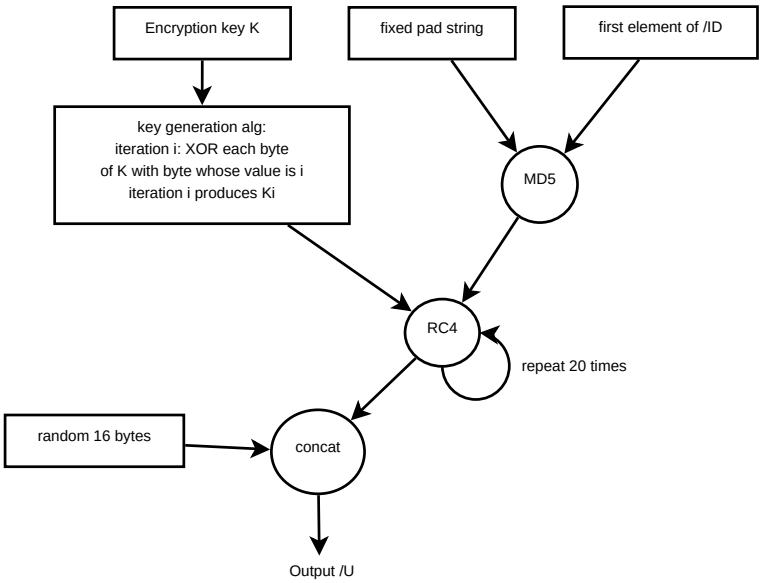

Figure 4: Algorithm to produce `/U`.

There are many known attacks on both RC4 and MD5 used in algorithms used for key generation (see [2] and [3], respectively). However, since these routines are used many times, further research is needed.

4 Conclusion

Basic options of PDF encryption were presented. At the moment proof-of-concept script is still work in progress.

References

[1] PDF Reference, fourth edition: Adobe Portable Document Format version 1.5.

[2] Klein, Andreas. "Attacks on the RC4 stream cipher." *Designs, Codes and Cryptography* 48.3 (2008): 269-286.

[3] Wang, Xiaoyun, and Hongbo Yu. "How to break MD5 and other hash functions." *Advances in Cryptology–EUROCRYPT 2005*. Springer Berlin Heidelberg, 2005. 19-35.