Digital Signatures and p7m files

Fabio Massimo Ercoli *

November 15st 2024

1 Introduction

We start from a given file named *Log 2024 CS - 2024 syllabus.pdf.p7m* [1] having the extension *p7m*.

The extension *p7m* is usually used to denote files that envelop a signed file, a signature or a series of signatures and a certificate or a list of the certificates related to the signatures.

In general, different signing techniques are used to sign and verify the signatures. For instance PAdES is used for human-signed PDF files, or CAdES for generic binary human-signed files, XAdES for software-signed XML files and JAdES for software-signed JSON files.

For some options, like CAdES for instance, when more signatures are applied, it is possibile to decide to sign the signed document signed by the last signature, doing what is called parallel signature, or to sign the entire envelope, that is a matrioska signature. For other options like PAdES only parallel signatures are available, and this may simplify a lot the extraction of the original document.

2 Extract the orginal document

For extracting the original signed file I searched for a non invasive solution, so I followed the idea of trying some online tool.

After a series of attempts I found the website for a commercial tool [2] offering the possibility to extract the original file from a p7m envelope file as a preview.

The result is the file named *Log 2024 CS - 2024 syllabus.pdf.pdf* that I attached it to the delivery directory.

The file is an openable fully working PDF containing what is claimed by the file name. So I think it worked. Later we will see how to do it with OpenSSL.

*More on me: https://github.com/fax4ever https://www.linkedin.com/in/fabioercoli/.

[1] The original file 'Log 2024 CS - 2024 syllabus.pdf.p7m' can be found on the delivery directory together with the homework PDF and LaTeX source file.

[2] https://www.coolutils.com/online/P7M-to-PDF

3 Show the certicate

Using my operating system Fedora Linux, If I try to open the p7m file directly with the file navigator by clicking to *detail*, I got the information I put in the plaintext file *cert-sign-non-repudiation.txt*. I attached it to the delivery directory as well. Later we will see a better way to extract these information.

Let's try to see what the file contains.

Opening it, it looks like a X.509 certificate, in particular a version 3 one. In there we can found all the fields we expect to find, according to the standard:

- **Version**: as we said it is a 3 version certificate.
- **Serial Number**: *22 F7 2A*, that is unique in the context of the same issuer (CA).
- **Signature**: this is the last field, since it signs all the precedent fields. It also contains the algorithm used *1.2.840.113549.1.1.11* and the parameter passed to it *05 00*.
- **Issuer Name**: the X.500 name of the issuing CA, Telecom Italia Trust Technologies S.r.l.
- **Validity**: consisting in two fields, *Not Valid Before* having value *2023-04-03* and *Not Valid After* having value *2026-04-03*.
- **Subject Name**: the X.500 name of the entity whose key is being certified.
- **Public Key Info**: that is the public key certified by the certificate.
- **Key Usage**: this parameter describes the goal of the use of the public key, in this case **no repudiation**.
- **Extension**: there are several extention fields, they look like references to other certificates, possibly enabling a verification chain up to a root (self-signed) CA, but I'm not sure about that.

The values of some fields, for instance the serial number and the parameter, used in the signed signature, are expressed as a sequence of 2 exadecimal values, so each element of the sequence is a 32-bit value. This representation allows to express a binary value in a text portable format.

4 Working with OpenSSL

4.1 Extract the certificate

First of all, I renamed *Log 2024 CS - 2024 syllabus.pdf.p7m* file using a simple name like *smime.p7m*. This step is of course not necessary but it will make the next OpenSSL command easier to read.
Now we try to extract the signer certicate using the command:

```
$ openssl pkcs7 -in smime.p7m -inform DER \
  -print_certs -text > senders-cert.pem
```

Opening the produced file *senders-cert.pem* with the operating system, we will have the same result reported in the previous chapter. But now we have extracted the certificate from the original file.

Moreover, if now we open the file with a text editor, all the contents are clear. For instance, I realized that the assumption I made on the extension was wrong. The extensions are very clear now, thanks to the *-text* option of OpenSSL, for instance we can read the certificate policies and the key usage, that is for non repudiation tougher with other information.

The extension also contains the url *http://ca.tipki.it/TTQTSPCA1/CERT*. Opening the url a self signed certicate can be downloaded. It is self signed because the *Subject Name* and *Issuer Name* are the same.

4.2 Get the original plaintext, verifying the signature

It is possible to extract the original signed document using the command:

```
$ openssl smime -verify -inform DER -in smime.p7m \
  -noverify -signer senders-cert.pem -out bla.pdf
```

The output is:

```
Verification successful
```

The produced file *bla.pdf* is the original plain text. The pdf is readable and it is identical to the one we extracted using the web tool in a previous section.

Notice that we used the *-inform DER* option to allow OpenSSL to work with DER files, that is not the default of OpenSSL.

The option *-verify* means that the signature has been verified using the public key provided by the certificate. But with the option *-noverify* we didn't verify signers certificate.
If we remove *-noverify* option we get the error:

```
Verification failure
8052A76B6A7F0000:error:10800075:PKCS7 routines:PKCS7_verify:certificate verify error:
crypto/pkcs7/pk7_smime.c:296:Verify error: unable to get local issuer certificate
```

According to the message it seems that it is not possible to verify the issuer of the certificate, in this case *Telecom Italia Trust Technologies S.r.l.*. I'm not sure that the p7m file contains all the certificate chains. I tried to extract more certificates from it, but I failed.