Digital Signatures and p7m files

Fabio Massimo Ercoli *

November 30st 2024

1 Goals

A *.p7m* file is given, it is produced by *CieSign* application. We want to:

- Analyze the file using *OpenSSL*
- Try to understand why the signature verification fails. What is missing?

Note I renamed the file *eIDAS Login_ secure online access to EU public services.pdf.p7m* to *my-file.p7m* to make the commands more readable.

2 Analyze the p7m file

Open the file with the operating system default application I discovered that more certificates are present. This is very common since we usually have a chain of certificates, starting from the CA (Certification Authority) of the signer of the document up to a level 1 CA or up to a CA that is recognized by the operating system.

To extract the certificates I run the following command:

```
openssl pkcs7 -in my-file.p7m -inform DER -print_certs
  -text > certificates.txt
```

Opening the *certificates.txt* file we have three X.509 version 3 certificates. They form a chain of certificates.

2.1 Professor's certificate

This is the certificate that contains the public key corresponding to the private key used to sign the signed document, so the public key that is supposed to used to verify the signature of the signed document. The public key is contained in the section *Subject Public Key Info*.

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

The *Validity* says that the certificate is not expired, but in order to be valid we need to check if it hasn't been revoked.

In the *Subject* we found the identity of signer, in this case the identity provided by the Italian Electronic Identity Card of Professor D'Amore. In the *Issuer* field we found the following X.500 name:

```
CN=Issuing sub CA for the Italian Electronic Identity Card - SUBCA002,
OU=Direz. Centr. per i Servizi Demografici - CNSD,
O=Ministero dell'Interno, C=IT
```

As its name suggests this is a sub-CA, so a low level CA providing certificates directly to the final users. The sign contained in this certificate is made using the private key of this entity and can be verified using the corresponding public key.

In the extensions we can find some references to the CA identities, links and the emphX509v3 Key Usage, in this case the only option is *Digital Signature*, It means that the subject is supposed to sign documents but not certificates.

2.2 Sub CA certificate

The public key of the sub CA is contained in (and certificate by) another certificate. According to its *Validity* field it is not expired. While it could be not valid, since the certificate does not provide any information about revocations. The issue is an higher level CA, this is the X.500 name:

```
CN=National root CA for the Italian Electronic Identity Card,
OU=Direz. Centr. per i Servizi Demografici - CNSD,
O=Ministero dell'Interno, C=IT
```

We notice that the issuer comes from the same organizational unit of the subject. Moreover, we can notice looking at the extensions that the subject can use his private key not only to sign documents but also to sign certificates.

This certificate was signed using the root CA private key and can be verified using the root CA public key.

2.3 Root CA certificate

Finally, the last certificate contains the public key of the root CA, since the subject is the root CA. Who is the issuer? The issuer is again the root CA.

This kind of certificate is called *self-signed*. Usually a self-signed certificate denotes the end of the certificate chain, in which we found the root.

The sign of a self-signed certificate does not provide any security value by itself. It means that the system must already know that the public key of the subject of a self-signed certificate. Usually this happens for level 1 CA.

3 Sign verification

3.1 Verify message signature only

As a first step we verify the sign on the document, without verifying the signs on the certificates. Using the following OpenSSL command:

```
openssl smime -verify -inform DER -in my-file.p7m
  -noverify -out my-file.pdf
```

The result is:

```
Verification successful
```

We used the option *-noverify*, but not *-nosigs*. According to the help of the OpenSSL verify command we have:

```
-noverify              Don't verify signers certificate
-nosigs                Don't verify message signature
```

This means that we have verified the message signature, but not signers certificate. Also we extracted the signed message: *my-file.pdf*. The can be opened and describes features about eIDAS regulation.

3.2 Verify message signature and signers certificate

Removing the option *-noverify* the command:

```
openssl smime -verify -inform DER -in my-file.p7m
  -out my-file.pdf
```

Has a different outcome:

```
Verification failure
8062E2747D7F0000:error:10800075:PKCS7 routines:PKCS7_verify:certificate
verify error:crypto/pkcs7/pk7_smime.c:296:Verify error:
 self-signed certificate in certificate chain
```

The reason of the failure is the fact that we have a self-signed certificate in the certificate chain (we already know that) and the subject (and his public key) is not already recognized by the system.

Starting from the *certificates.txt* file we produced in the previous chapters containing the three certificates, I created a new file containing only the self-signed certificate, naming it *rootca.pem*.

Trying to apply the option *-CAFile* to make it trusted with the command:

```
openssl smime -verify -inform DER -in my-file.p7m
  -CAfile ./rootca.pem
```

I got now a new error:

```
Verification failure
80A23585137F0000:error:10800075:PKCS7
routines:PKCS7_verify:certificate
verify error:crypto/pkcs7/pk7_smime.c:296:Verify
error: unsuitable certificate purpose
```

3.3 Using an online tool to verify the p7m file

Exploring the case I tried to use an online tool[1] to verify the p7m file.

The tool reported several errors (translated from the Italian):

- The certificate issuer is not valid for the selected validation profile
- The certificate does not comply with Regulation (EU) No. 910/2014 - eIDAS and its amendments and additions
- dnQualifier not present
- CA Issuer not present

4 What is missing?

Trying to use the same OpenSSL command on the last homework p7m file, using the command:

```
openssl smime -verify -inform DER -in
  'Log-2024-CS---2024-syllabus.pdf.p7m'
  -out bla.pdf -CAfile ./CERT.cer
```

Once we downloaded and put in the same folder the self signed root certificate *CERT.cer*, everything works:

```
Verification successful
```

So would suggest that there is something missing in the current homework certificates that were present in the last homework certificates.

Looking at the OpenSSL documentation, in particular to the verification options page[2], there is a *-purpose* option.

Redoing the sign verification with *-purpose any*:

```
openssl smime -verify -inform DER -in my_file.p7m
  -CAfile ./rootca.pem -purpose any
```

this time works:

```
Verification successful
```

[1] https://arubasign.arubapec.it/asonline/

[2] https //docs.openssl.org/3.0/man1/openssl-verification-options/#verification-options

It seems that if no *-purpose* option is passed, the *-purpose smimesign* value is applied by default. According to a stack overflow question [3] it seems that if the extended key usages are defined, they must contain the *emailProtection* value in order to match the *smimesign* purpose.

We can confirm it using *-purpose* function of OpenSSL x509 utility. If we put the user certificate, the one that contains the public key of the identity used to sign the document, in a separate file, for instance *user.pem* running the command:

```
openssl x509 -in ./user.pem -noout -purpose
```

We can see that the certificate does not have the purpose of *S/MIME signing*:

```
Certificate purposes:
SSL client : Yes
SSL client CA : No
SSL server : No
SSL server CA : No
Netscape SSL server : No
Netscape SSL server CA : No
S/MIME signing : No
S/MIME signing CA : No
S/MIME encryption : No
S/MIME encryption CA : No
CRL signing : No
CRL signing CA : No
Any Purpose : Yes
Any Purpose CA : Yes
OCSP helper : Yes
OCSP helper CA : No
Time Stamp signing : No
Time Stamp signing CA : No
Code signing : No
Code signing CA : No
```

So my guess this is what is missing from the user certificate.

[3] https://stackoverflow.com/questions/40685111/why-openssls-pkcs7-verify-requires-smimesign-certificate-purpose