HW04

Riccardo Zucchelli 1984963

October 2024

1 Introduction

The homework is based on analysing, extracting and verifying the contents of .p7m file. A .p7m is the filename extension found in digitally signed message, and it's part of the `PKCS #7` standard, developed by RSA Security. It's mostly found in digitally signed emails that uses MIME, like we find in the Italian system Posta Elettronica Certificata, a common tool used to send message to Public Administration, but it's use is not limited to this.

1.1 What does it mean to sign a message?

Signing a message is the digital equivalent of endorsing a paper document, adding multiple specifications to it, since it tells the other party that you reviewed it and agreed on its content.

Message signing is part of a bigger topic on **Authentication**, based on proving the identity that someone is presenting, and also ensuring that, once signed, an object is uniquely associated to the signee identity, and that its contents have not been tampered. Its use can be seen throughout the Internet, from signing important legal document to ensuring a secure connection to a website.

1.2 Document signing

There are different standards for signing a document, but one of the most important ones are based on **AdES**, or **Advanced digital Electronic Signature**

1. **PAdES**: Although anything can be signed using a standardized file, PDF have been popular enough to have a standard built in itself to support digital signatures. The P stands for PDF, and all informations are added to the document itself. It guarantees data integrity, but not visual integrity, since any dynamic content added may be rendered differently.

2. **CAdES**: This is the what everyone should use for signing any generic content that hasn't a more specific AdES standard. This doesn't mean that someone couldn't for example sign a PDF with this, but it wouldn't be efficient.

2 The Homework

We have a .p7m file (with filename `Log 2024 CS - 2024 syllabus.pdf.p7m`) that has been signed with a modern standard, and we want to do the following

1. Extract plaintext.
2. Verify file integrity.
3. Show the details of the certificates, along with the signer identity and eventual policies of the certificate.

To do this we'll use the libraries included in OpenSSL, which is a software library that provides a free set of encryption/signing tools. We'll be more interested in the latter ones.

2.1 Getting more infos on the file

Using the OpenSSL available online we can try and find some indications on how to start working on this. As a start, we may want to try learning more about the `openssl smime`, since as previously stated this is a very well known usage of document signing.

Trying with the command `openssl smime -verify -in file.p7m -out content.pdf`, that loosely follows OpenSSL standards on how to input and output files, results with an error:

```
40A7BE3E67710000:error:068000D1:asn1 encoding
routines:SMIME\_read\_ASN1\_ex:
no content type:../crypto/asn1/asn\_mime.c:415:
```

This points us to the right way: the default content type (which why loading the documentation tells us is **SMIME**) has not been found. We'll need to add it more information on the type, which may be DER or PEM. Trying again, now trying first with DER, `openssl smime -verify -in file.p7m -inform DER -out content.pdf` results in

```
    Verification failure
4077F43FE4790000:error:10800075:PKCS7 routines:PKCS7_verify:
certificate verify error:../crypto/pkcs7/pk7_smime.c:293:
Verify error: unable to get local issuer certificate
```

which means that the file has been correctly recognized, but given that the certificate issuer hasn't been extracted the verification failed. We won't verify the signers certificate, but just the signed message contents. This will add the parameter `-noverify` to our command. Using that ulterior parameter we get as output `Verification successful` and now, in our folder, we have content.pdf that has been verified as untouched! We can also extract the certificate for further tests with the `-signer` directive.

2.2 What about the certificate?

Now that we have a certificate with which we can work, getting more informations about the identity of the signer is easy. Using always OpenSSL `x509` set of tools to input our certificate **cert.pem**, and to print out the certificate information in a text form.

```
HW04 openssl x509 -in cert.pem -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 2291498 (0x22f72a)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = IT, O = Telecom Italia Trust Technologies S.r.l.,
            OU = Qualified Trust Service Provider,
            CN = TI Trust Technologies QTSP CA 1,
            organizationIdentifier = VATIT-04599340967
        Validity
            Not Before: Apr  3 20:38:58 2023 GMT
            Not After : Apr  3 20:38:58 2026 GMT
        Subject: C = IT, CN = FABRIZIO D'AMORE,
------------SN=D'AMORE, GN = FABRIZIO,
            serialNumber = TINIT-DMRFRZ60P04H501I,
            dnQualifier = TITT030423203857337,
            O = Universit\C3\A0 degli Studi di Roma \"La Sapienza\",
            organizationIdentifier = VATIT-80209930587
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:a1:c2:6f:ad:97:91:9c:fb:88:c2:f0:de:22:f8:
                    34:5c:77:76:56:9a:33:b1:a9:3e:2f:b4:e6:35:3e:
                    ab:37:60:6b:7d:09:6c:25:5f:3a:ba:77:a6:6f:ee:
                    e3:bf:8b:2d:16:ce:c5:39:5b:e9:18:16:6c:08:97:
                    63:a9:84:78:20:4d:ef:a9:e9:d5:55:c3:07:d0:ff:
                    54:3a:5c:9f:9e:72:35:9d:dc:40:7e:23:2f:0e:ab:
                    80:7f:f1:9e:7d:6a:4a:0e:51:74:25:92:0c:65:61:
                    31:ff:de:06:15:14:bf:01:11:d6:b0:f6:30:a5:6f:
                    f5:48:fe:a6:e0:b5:7e:6b:ad:b3:0d:a0:0a:15:01:
                    0b:c0:7a:bb:f6:1c:ef:94:19:c0:cc:0f:b8:dc:e9:
                    6b:c8:2d:19:15:7e:79:47:e6:22:a8:13:66:d4:f8:
                    58:d6:f1:46:c0:04:cd:80:7c:51:d9:99:21:1b:c4:
                    db:ce:54:be:40:89:df:9e:9c:e1:71:0b:3d:a4:bb:
                    90:5d:58:25:9b:e2:e5:bf:f7:02:d4:46:b9:26:c2:
                    26:6e:4a:5b:d0:99:fa:63:ab:52:ac:c8:8e:1d:ea:
                    1c:f1:24:9e:dd:2b:6b:64:59:be:06:f3:b8:ce:b7:
                    bb:0a:8e:29:71:3c:c3:ec:40:79:25:7a:d1:d9:6c:
```

```
                7a:f7
            Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Certificate Policies:
                Policy: 1.3.76.33.1.1.20
                  CPS: https://www.trusttechnologies.it/download/documentazione/
                  User Notice:
                    Explicit Text: The certificate holder must
                    use the certificate only for the purposes for
                    which it is issued.
                Policy: 0.4.0.194112.1.2
                Policy: 1.3.76.16.6
            qcStatements:
                0..0.....F..0.....F.....0.....F..0S.....F..0I0G.A
                https://www.trusttechnologies.it/download/disclosure-statement-q
            X509v3 Subject Alternative Name:
                othername: 1.3.6.1.4.1.1466.115.121.1.50::+393286762246
            X509v3 Key Usage: critical
                Non Repudiation
            X509v3 Authority Key Identifier:
                7D:87:2D:F8:1F:B3:B0:D4:B6:89:80:2C:B4:AA:CE:DE:3C:08:22:06
            X509v3 CRL Distribution Points:
                Full Name:
                  URI:http://ca.tipki.it/TTQTSPCA1/CDP10
            X509v3 Subject Key Identifier:
                5B:A0:E6:4A:00:8B:ED:D9:0D:4B:28:D5:15:5D:45:FA:93:24:80:17
            Authority Information Access:
                CA Issuers - URI:http://ca.tipki.it/TTQTSPCA1/CERT
                OCSP - URI:http://ocsp.tipki.it
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        92:9c:12:4f:55:cb:25:15:39:c4:a6:03:b1:80:b9:d8:e4:d9:
        9d:b5:ff:7e:a1:ca:f3:40:7c:a4:10:8a:7b:da:53:ca:4c:43:
        f2:f8:83:6c:cc:c2:44:7e:bf:1c:13:97:33:bb:5d:7e:54:12:
        d3:a8:81:f1:33:08:bd:da:2b:6d:4d:d7:f4:d0:20:39:5a:cf:
        94:96:b2:bc:eb:f7:4c:3a:ee:7a:04:e5:c0:27:b8:9e:fb:9c:
        c3:a3:3a:52:b2:b4:e7:a8:5b:35:58:71:83:4c:b9:ec:5c:a6:
        d0:09:b1:4f:c0:a2:aa:26:08:ef:cc:c1:4c:7f:9a:1c:49:8d:
        e1:26:b8:81:1e:2f:45:e5:ff:d1:9a:66:95:4c:57:58:4a:c2:
        5e:05:09:b1:e0:8d:87:e1:f6:e9:15:0c:9d:8b:0d:1c:e9:13:
        38:64:32:86:b3:ab:91:27:87:b6:ed:9b:36:0e:da:92:d6:6c:
        8a:99:a1:17:3d:75:40:05:2c:c9:ce:84:2e:b2:11:3d:fe:cb:
        e8:05:35:93:23:40:6e:8f:79:df:41:dd:f2:de:55:91:84:4a:
        b1:68:12:d9:be:9d:b3:f2:83:45:65:bd:0e:31:61:08:81:76:
        81:0d:9e:18:19:26:44:36:47:95:1a:c0:86:ed:27:0b:7c:ab:
        1b:e9:98:f1:4a:13:dd:d5:9b:c9:31:df:bd:7c:e3:fb:27:2e:
```

```
        00:47:9b:a0:4f:23:66:56:cc:9d:b0:1f:75:20:16:4f:bb:3d:
        c1:f1:8e:65:a4:7e:65:c5:52:5d:44:e3:2e:9a:29:8e:6a:0d:
        9d:53:cc:3c:fc:4d:42:58:1e:e0:bc:56:9c:5b:07:fc:f4:0e:
        7f:b4:de:15:95:46:1b:59:22:98:9d:7a:d9:53:7e:34:f5:2c:
        73:55:8f:88:f8:dc:1e:49:e1:38:d9:46:4c:27:94:6f:ed:97:
        de:2a:ca:05:d6:59:34:d7:2e:8b:19:3d:59:dd:f0:53:eb:51:
        b0:f5:bd:d8:9c:98:88:1c:95:98:a6:d2:6e:bb:22:ba:46:06:
        69:8d:cc:ba:4a:88:82:4c:f6:b6:c0:f0:02:ef:49:23:13:d2:
        fb:df:85:23:e7:12:f4:df:60:ef:5f:f8:c5:58:a3:88:d2:ce:
        ad:b0:32:02:ff:0f:2c:94:f8:79:29:ef:45:e0:f7:45:a6:6d:
        5b:a1:5e:8d:e4:97:29:91:53:39:fd:a7:86:b8:2b:4f:d1:34:
        e1:05:a2:2a:a9:16:4e:1e:ff:7b:b4:7a:4b:1b:b4:0f:d7:8c:
        f3:ed:3b:5c:0d:16:f7:05:2f:8a:b4:eb:d4:5d:ae:1d:19:9e:
        e4:db:be:4d:f8:80:33:ed
-----BEGIN CERTIFICATE-----
[Certificate value in PEM format, removed]
-----END CERTIFICATE-----
```

This command prints out all the information about the certificate that it contains. The details include essential data regarding the subject, issuer, and validity period of the certificate, among other critical attributes. In the first part, we can find three key values:

- **Subject**: This section identifies the entity to which the certificate has been issued. It typically includes fields such as the Common Name (CN), Organization (O), Organizational Unit (OU), Country (C), and, in this case, also an organizationalIdentifier. With this information we can find that Fabrizio D'Amore is the subject that signed the document first presented to us, and that it has signed it as a member of La Sapienza. We'll talk more about its valid uses later.

- **Issuer**: This field contains details about the Certificate Authority (CA) that issued the certificate. It includes similar fields as the Subject (like CN, O, and C). The issuer is responsible for validating the subject's identity and signing the certificate. With this information we can find that Telecom Italia Trust is the Certificate Authority that has given out this certificate. Since we know that the subject has also an organization in its fields, we can assume (and easily verify) that La Sapienza uses Telecom Italia Trust as its provider for the certificates issued for its professors.

- **Validity**: This segment outlines the period during which the certificate is valid. It consists of two dates:

  - **Not Before**: The date and time from which the certificate becomes valid.
  - **Not After**: The date and time when the certificate expires.

When examining the certificate with the openssl x509 command, we can find out that the basic certificate structure has been extended, by providing more attributes that supply further constraints and capabilities for the certificate's usage, enhancing its security and specificity. One of the most noteworthy attributes to be discussed are:

- **X.509v3 Certificate Policies**: These specify the rules under which the certificate was issued and how it can be used. Under `Explicit Text`, we find the statement:

  > *"The certificate holder must use the certificate only for the purposes for which it is issued."*

  This means that if the certificate owner were to use this certificate to sign a personal document, it would not be considered valid, as it would fall outside the intended use specified by the certificate's policy. This stresses the importance of manually verifying a certificate, since no automatic process will ever be able to check for a policy!

- **X.509v3 Key Usage**: Specifies what's the purpose of the public key found within the certificate. Marked as `critical`, the **Non Repudiation** value tells it that the document's signature can't be denied in a court of law.

- **Authority Information Access**: Specifies the ways to access information about the issuer, such as the issuer's certificate or Online Certificate Status Protocol (OCSP) responder. The latter protocol is an essential way to verify that a certificate hasn't been revoken, and an upgrade over the previous Certificate Revokation List, which usually takes two weeks to update and may be too long of a time to take appropriate security measures. We can use the`CA Issuers: http://ca.tipki.it/TTQTSPCA1/CERT` to verify the certificate validity, using the CA certificate.

Downloading the certificate (and converting it to a PEM file) makes it possible to verify that Fabrizio D'Amore's certificate is valid, using `openssl verify`. Giving as a command `openssl verify CAfile CAcert.pem crl_download -crl_check policy_check cert.pem` tells us that the certificate used on the original p7m file has been signed by the certificate authority, and since we manually verified that the policy is respected we can confidently say that the identity of the signer is confirmed.