Lab 4 - Digital Signatures

1. Key Generation with `openssl`

Outputs a text encoding of an RSA key-pair suitable both for RSA encryption and digital signature, and stores it into a PEM format file:

```
openssl genpkey -algorithm rsa > myRSApkey.pem
```

Produces a file containing the public parameters corresponding to one NIST standard elliptic curve called "prime256v1" or simply "P-256":

```
openssl genpkey -genparam -algorithm ec -pkeyopt ec_paramgen_curve:P-256 >
myECppar.pem
```

A key-pair for key agreement or digital signature can be generated from the public parameters with:

```
openssl genpkey -paramfile myECppar.pem > myECpkey.pem
```

Public key needs to be detached from the key-pair files with:

```
openssl pkey -pubout -in myECpkey.pem > myECpubkey.pem
```

2. Signature Generation and Verification

Generate a digital signature of a file using a key-pair:

```
openssl pkeyutl -inkey myECpkey.pem -sign -rawin -in <file> -out sign.bin
```

Previous signature can be verified from the corresponding public key with:

```
openssl pkeyutl -pubin -inkey myECpubkey.pem -verify -rawin -in <file> -sigfile
sig.bin
```

Since the key-pair file also contains the public key, verification can alternatively be done directly from it:

```
openssl pkeyutl -inkey myECpkey.pem -verify -rawin -in mydoc.doc -sigfile sig.bin
```

3. Certificate Management

A X.509 public key certificate is basically a document digitally signed by a certification authority (CA), that includes information about the certificate issuer (the CA), the subject identity and the subject's public key to be certified, among other things.

X.509 certificates are issued by a CA on request. Indeed, a subject must create certificate signature request (CSR) and send it to the CA. Then, the CA transforms the CSR into a certificate.

3.1 Obtaining a certificate from a CA

When an end user wants to obtain a public key certificate, first a CSR file must be created and submitted to the CA:

```
openssl req -new -key myECpkey.pem -out cert_req.pem
```

The contents of the generated CSR can be inspected with the command:

```
openssl req -in cert_req.pem -text
```

The following command does the conversion from the CSR to a certificate:

```
openssl x509 -in cert_req.pem -req -CA CAcert.pem -CAkey CApkey.pem -out mycert.pem
```

- `CApkey.pem` is the CA key-pair
- `CAcert.pem` is the CA public key certificate

The certificate contents can be inspected with:

```
openssl x509 -in mycert.pem -text
```

3.2 Certificate validation

A public key certificate can be validated by verifying all the signatures contained in all certificates in the trust chain.

The following command checks the validity of a peer's certificate:

```
openssl verify -CAfile CAcert.pem -check_ss_sig peer_cert.pem
```

> **Info**
>
> For longer certificate chains (with intermediate CAs) the intermediate CA certificates must be aggregated into a single file and provided to the `openssl` call with a special option (see the documentation for openssl verify)

Generation of self signed CA certificates:

```
openssl req -x509 -new -key myECpkey.pem -out myselfCAcert.pem
```

Certificate public key can be extracted with:

```
openssl x509 -in mycert.pem -pubkey -noout > extractedpubkey.pem
```

The certificate file can replace the public key file in the signature verification:

```
openssl pkeyutl -certin -inkey mycert.pem -verify -rawin -in mydoc.doc -sigfile
sig.bin
```

> ⚠ **Warning**
>
> For security reasons, before any use of a received peer's public key, the corresponding certificate chain must be verified as explained above.

3.3 Certificate Revocation

Certificates can be revoked for many reasons before their expiration date. For this reason, the public key certificate verification process must ensure that none of the certificates in the chain has been revoked. CA usually publishes Certificate Revocation Lists (CRL) in a periodic basis.

A CRL is mainly a list of revoked certificates signed by the issuing CA.

The user must always use the CRL to check the revocation status of all certificates in a certificate chain before using a peer's public key.

Inspect a CRL file:

```
openssl crl -in crl001.pem -text
```

The CRL file itself can be verified with:

```
openssl crl -in crl001.pem -CAfile CAcert.pem -noout
```

The complete verification of a peer's certificate taking into account the CRL is:

```
openssl verify -CAfile CAcert.pem -check_ss_sig -CRLfile crl001.pem -crl_check
peercert.pem
```

> ⚠ **Warning**

If there are some intermediate CAs involved in the verification, their certificates must also be checked for possible revocations using `-crl_check_all` instead of `-crl_check`

4. Practical work

Instructions:

1. Create a public parameters file for NIST elliptic curve P-256, and generate a key pair to be used for ECDSA signatures
2. Create a CSR for the generated key pairs
3. Send to the CA the CSR
4. Download public key root CA certificate, the CRL lists and all the other certificate document and signature files
5. Verify the validity of all this material and write a report with the details of all verification done
6. Digitally sign the report

4.1 Generate P-256 ECDSA keys

Generate parameter file with P-256 EC:

```
openssl genpkey -genparam -algorithm ec -pkeyopt ec_paramgen_curve:P-256 >
myECppar.pem
```

Generate key-pair from parameter file:

```
openssl genpkey -paramfile myECppar.pem > myECpkey.pem
```

Extract public key from key-pair file:

```
openssl pkey -pubout -in myECpkey.pem > myECpubkey.pem
```

4.2 Generate CSR

Generate a certificate request using our key-pair:

```
$ openssl req -new -key myECpkey.pem -out cert_req.pem
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ES
State or Province Name (full name) [Some-State]:Barcelona
```

```
Locality Name (eg, city) []:Barcelona
Organization Name (eg, company) [Internet Widgits Pty Ltd]:UPC
Organizational Unit Name (eg, section) []:DPROT
Common Name (e.g. server FQDN or YOUR name) []:Student31
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

4.3 Get Certificate

Content of the certificate delivered by the CA after sending our CSR:

```
$ cat 102_cert.pem
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 16803433828964055688 (0xe931ca09fe2f3a88)
    Signature Algorithm: ecdsa-with-SHA256
        Issuer: C=ES, ST=Barcelona, L=Barcelona, O=UPC, OU=MAK,
CN=Jorge/emailAddress=jorge.villar@upc.edu
        Validity
            Not Before: Dec 27 09:20:52 2023 GMT
            Not After : Dec 26 09:20:52 2024 GMT
        Subject: C=ES, ST=Barcelona, O=UPC, OU=DPROT, CN=Student31
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:50:10:ef:43:95:3b:c8:2d:2a:fa:60:10:27:62:
                    aa:4f:a6:ac:0c:a3:42:2f:72:1b:d1:47:e0:4e:0a:
                    6c:b2:9d:ea:49:ea:21:e5:55:7a:00:21:1c:40:2f:
                    31:82:fc:53:27:3e:26:20:12:80:1f:64:5d:bd:ab:
                    09:5f:dc:50:a4
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                C0:2F:CC:24:0A:3F:D2:E6:F6:14:B1:E3:E2:B8:91:33:F4:26:9B:F7
            X509v3 Authority Key Identifier:
                keyid:F5:3C:59:F9:A4:FA:CD:C3:0D:A2:D4:12:BA:5A:F9:41:EA:19:48:2C

    Signature Algorithm: ecdsa-with-SHA256
         30:45:02:20:69:a5:32:31:08:58:d8:a3:86:99:81:97:ed:da:
         d3:55:f8:0d:10:92:80:9d:ae:00:2a:da:60:69:cf:a3:52:f9:
         02:21:00:a5:1c:6f:34:52:9b:8e:c5:d5:f7:e2:11:f1:4e:8f:
```

```
        47:0d:d8:f2:cc:19:59:61:3d:e5:3a:1d:7a:cd:e7:21:31
-----BEGIN CERTIFICATE-----
MIICTDCCAfKgAwIBAgIJAOkxygn+LzqIMAoGCCqGSM49BAMCMIGGMQswCQYDVQQG
EwJFUzESMBAGA1UECAwJQmFyY2Vsb25hMRIwEAYDVQQHDAlCYXJjZWxvbmExDDAK
BgNVBAoMA1VQQzEMMAoGA1UECwwDTUFLMQ4wDAYDVQQDDAVKb3JnZTEjMCEGCSqG
SIb3DQEJARYUam9yZ2UudmlsbGFyQHVwYy5lZHUwHhcNMjMxMjI3MDkyMDUyWhcN
MjQxMjI2MDkyMDUyWjBTMQswCQYDVQQGEwJFUzESMBAGA1UECAwJQmFyY2Vsb25h
MQwwCgYDVQQKDANVUEMxDjAMBgNVBAsMBURQUk9UMRIwEAYDVQQDDAlTdHVkZW50
MzEwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAARQEO9DlTvILSr6YBAnYqpPpqwM
o0IvchvRR+BOCmyynepJ6iHlVXoAIRxALzGC/FMnPiYgEoAfZF29qwlf3FCko3sw
eTAJBgNVHRMEAjAAMCwGCWCGSAGG+EIBDQQfFh1PcGVuU1NMIEdlbmVyYXRlZCBD
ZXJ0aWZpY2F0ZTAdBgNVHQ4EFgQUwC/MJAo/0ub2FLHj4riRM/Qmm/cwHwYDVR0j
BBgwFoAU9TxZ+aT6zcMNotQSulr5QeoZSCwwCgYIKoZIzj0EAwIDSAAwRQIgaaUy
MQhY2KOGmYGX7drTVfgNEJKAna4AKtpgac+jUvkCIQClHG80UpuOxdX34hHxTo9H
DdjyzBlZYT3lOh16zechMQ==
-----END CERTIFICATE-----
```

4.4 Download files

Files downloaded:

```
files/
├── 1
│   ├── README.TXT
│   ├── albert_cert.pem
│   ├── crl_001.pem
│   ├── e_mc2.pdf
│   └── sig_001.b64
├── 2
│   ├── README.TXT
│   ├── crl_002.pem
│   ├── flat_e.pdf
│   ├── isaac_cert.pem
│   └── sig_002.b64
├── 3
│   ├── README.TXT
│   ├── crl_002b.pem
│   ├── g_dice.pdf
│   ├── niels_cert.pem
│   └── sig_003.b64
├── P-256.pem
├── crl_003.pem
└── rootCAcert.pem
```

4.5 Validate files

4.5.1 Albert

4.5.1.1 Check certificate authenticity

Albert's certificate:

```
$ openssl x509 -in albert_cert.pem -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            e9:31:ca:09:fe:2f:3a:7c
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: C = ES, ST = Barcelona, L = Barcelona, O = UPC, OU = MAK, CN = Jorge,
emailAddress = jorge.villar@upc.edu
        Validity
            Not Before: Nov 27 18:38:33 2023 GMT
            Not After : Nov 26 18:38:33 2024 GMT
        Subject: C = ES, ST = Barcelona, O = UPC, CN = Albert Einstein, emailAddress =
aeinstein@upc.edu
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:5c:8a:cb:83:17:33:d7:78:4d:d8:49:39:aa:d4:
                    18:7a:84:80:0a:0c:5c:f0:9d:ee:ac:8c:d8:1f:d9:
                    eb:65:4e:46:2a:e9:09:0b:a6:1d:54:7a:fb:d5:c3:
                    a9:32:e1:2f:f9:61:da:a4:f6:56:99:6c:29:09:33:
                    d3:de:76:91:ca
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                B4:64:7E:C1:A5:3F:41:14:89:B2:96:CA:53:A0:37:2C:C5:58:35:02
            X509v3 Authority Key Identifier:
                F5:3C:59:F9:A4:FA:CD:C3:0D:A2:D4:12:BA:5A:F9:41:EA:19:48:2C
    Signature Algorithm: ecdsa-with-SHA256
    Signature Value:
        30:45:02:21:00:e8:0b:8d:e2:31:e7:aa:1d:02:92:5f:b7:ed:
        6c:c3:43:02:50:3e:6c:89:8a:05:88:a7:62:26:7f:02:37:d3:
        65:02:20:1a:93:b9:45:4b:f1:40:b3:25:f5:13:2f:4b:d1:91:
        6a:54:57:24:72:81:36:4d:c8:1c:6b:5d:fa:93:05:1c:67
-----BEGIN CERTIFICATE-----
MIICZDCCAgqgAwIBAgIJAOkxygn+Lzp8MAoGCCqGSM49BAMCMIGGMQswCQYDVQQG
EwJFUzESMBAGA1UECAwJQmFyY2Vsb25hMRIwEAYDVQQHDAlCYXJjZWxvbmExDDAK
BgNVBAoMA1VQQzEMMAoGA1UECwwDTUFLMQ4wDAYDVQQDDAVKb3JnZTEjMCEGCSqG
SIb3DQEJARYUam9yZ2UudmlsbGFyQHVwYy5lZHUwHhcNMjMxMTI3MTgzODMzWhcN
MjQxMTI2MTgzODMzWjBrMQswCQYDVQQGEwJFUzESMBAGA1UECAwJQmFyY2Vsb25h
MQwwCgYDVQQKDANVUEMxGDAWBgNVBAMMD0FsYmVydCBFaW5zdGVpbjEgMB4GCSqG
SIb3DQEJARYRYWVpbnN0ZWluQHVwYy5lZHUwWTATBgcqhkjOPQIBBggqhkjOPQMB
```

```
BwNCAARcisuDFzPXeE3YSTmq1Bh6hIAKDFzwne6sjNgf2etlTkYq6QkLph1UevvV
w6ky4S/5Ydqk9laZbCkJM9PedpHKo3sweTAJBgNVHRMEAjAAMCwGCWCGSAGG+EIB
DQQfFh1PcGVuU1NMIEdlbmVyYXRlZCBDZXJ0aWZpY2F0ZTAdBgNVHQ4EFgQUtGR+
waU/QRSJspbKU6A3LMVYNQIwHwYDVR0jBBgwFoAU9TxZ+aT6zcMNotQSulr5QeoZ
SCwwCgYIKoZIzj0EAwIDSAAwRQIhAOgLjeIx56odApJft+1sw0MCUD5siYoFiKdi
Jn8CN9NlAiAak7lFS/FAsyX1Ey9L0ZFqVFckcoE2Tcgca136kwUcZw==
-----END CERTIFICATE-----
```

Verify if the certificate is issued by our CA:

```
$ openssl verify -CAfile rootCAcert.pem albert_cert.pem
albert_cert.pem: OK
```

✓ **Valid**

The certificate was issued by our CA!

4.5.1.2 Check CRL validity

CRL content:

```
$ openssl crl -in crl_001.pem -noout -text
Certificate Revocation List (CRL):
        Version 2 (0x1)
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: C = ES, ST = Barcelona, L = Barcelona, O = UPC, OU = MAK, CN = Jorge,
emailAddress = jorge.villar@upc.edu
        Last Update: Nov 27 18:58:07 2023 GMT
        Next Update: Dec 27 18:58:07 2023 GMT
        CRL extensions:
            X509v3 CRL Number:
                3
Revoked Certificates:
    Serial Number: E931CA09FE2F3A7B
        Revocation Date: Nov 16 16:37:13 2023 GMT
        CRL entry extensions:
            X509v3 CRL Reason Code:
                Key Compromise
    Signature Algorithm: ecdsa-with-SHA256
    Signature Value:
        30:45:02:20:1e:ca:66:39:18:37:59:6a:fb:71:64:e8:4d:c9:
        a4:74:f0:6d:cb:d3:e9:45:f0:97:fe:a1:43:64:39:bd:f1:64:
        02:21:00:dd:bb:33:23:13:a9:ed:4e:40:b5:ae:3e:24:90:67:
        7c:d9:3e:bd:2e:e5:77:39:58:47:52:8b:78:e1:72:22:5b
```

Verify it is issued by our CA:

```
$ openssl crl -in crl_001.pem -CAfile rootCAcert.pem -noout -verify
verify OK
```

✓ **Valid**

The CRL was issued by our CA.

4.5.1.3 Check certificate revocation

In the `crl_001.pem` there is only one revoked certificate:

```
...
Revoked Certificates:
    Serial Number: E931CA09FE2F3A7B
...
```

Let's compare this serial number with the public certificate we have:

```
$ openssl x509 -in albert_cert.pem -serial -noout
serial=E931CA09FE2F3A7C
```

✓ **Valid**

Last digits are different so Albert's certificate is valid!

4.5.1.4 Check signature validity

First decode the signature:

```
cat sig_001.b64 | openssl enc -d -a > sig1.bin
```

Content of `sig1.bin`:

```
$ cat sig1.bin
0D RF%斫T(����T
��ARl@Q["  ��R4'n j w������Ml  ��  m  96%
```

Then verify it:

```
$ openssl pkeyutl -certin -inkey albert_cert.pem -verify -rawin -in e_mc2.pdf -sigfile
sig1.bin
Signature Verified Successfully
```

✓ **Valid**

Signature is valid: the pdf was not altered!

4.5.2 Isaac

4.5.2.1 Check signature validity

Isaac's certificate:

```
$ openssl x509 -in isaac_cert.pem -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            ee:37:c4:9f:49:ab:17:c3
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: C = ES, ST = Barcelona, L = Barcelona, O = UPC, CN = Sir Isaac Newton,
emailAddress = man.in.the.middle@hackers.com
        Validity
            Not Before: Nov 28 11:17:11 2023 GMT
            Not After : Dec 28 11:17:11 2023 GMT
        Subject: C = ES, ST = Barcelona, L = Barcelona, O = UPC, CN = Sir Isaac
Newton, emailAddress = man.in.the.middle@hackers.com
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:8e:6c:6c:b7:09:5b:1c:f7:b5:51:b3:b1:a1:7a:
                    fe:38:1e:7a:10:1f:b7:d5:c4:ff:f0:d5:a6:f8:6c:
                    91:ab:f0:b9:97:83:ca:07:5b:18:a8:f5:75:a0:eb:
                    20:a9:21:6a:5e:1c:21:4d:28:be:ed:87:12:f9:54:
                    96:9b:a4:11:38
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                FF:EF:7A:A3:0B:E7:23:E2:9C:EB:58:0C:41:11:56:29:A1:58:B4:C6
            X509v3 Authority Key Identifier:
                FF:EF:7A:A3:0B:E7:23:E2:9C:EB:58:0C:41:11:56:29:A1:58:B4:C6
            X509v3 Basic Constraints:
                CA:TRUE
    Signature Algorithm: ecdsa-with-SHA256
    Signature Value:
        30:45:02:21:00:bb:63:53:9a:21:06:2f:15:c1:3a:0f:d0:db:
        39:e8:3c:be:3f:4d:e0:b3:c6:d9:10:5e:af:a9:3d:58:10:06:
        51:02:20:64:15:6f:fa:6f:2c:6b:16:f6:d4:e3:24:2b:3c:40:
        02:3d:91:fc:3f:5b:fb:c0:e9:e6:5f:6e:80:c1:b6:75:4f
-----BEGIN CERTIFICATE-----
MIICYTCCAgegAwIBAgIJAO43xJ9JqxfDMAoGCCqGSM49BAMCMIGMMQswCQYDVQQG
```

```
EwJFUzESMBAGA1UECAwJQmFyY2Vsb25hMRIwEAYDVQQHDAlCYXJjZWxvbmExDDAK
BgNVBAoMA1VQQzEZMBcGA1UEAwwQU2lyIElzYWFjIE5ld3RvbjEsMCoGCSqGSIb3
DQEJARYdbWFuLmluLnRoZS5taWRkbGVAaGFja2Vycy5jb20wHhcNMjMxMTI4MTEx
NzExWhcNMjMxMjI4MTExNzExWjCBjDELMAkGA1UEBhMCRVMxEjAQBgNVBAgMCUJh
cmNlbG9uYTESMBAGA1UEBwwJQmFyY2Vsb25hMQwwCgYDVQQKDANVUEMxGTAXBgNV
BAMMEFNpciBJc2FhYyBOZXd0b24xLDAqBgkqhkiG9w0BCQEWHW1hbi5pbi50aGUu
bWlkZGxlQGhhY2tlcnMuY29tMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEjmxs
twlbHPe1UbOxoXr+OB56EB+31cT/8NWm+GyRq/C5l4PKB1sYqPV1oOsgqSFqXhwh
TSi+7YcS+VSWm6QROKNQME4wHQYDVR0OBBYEFP/veqML5yPinOtYDEERVimhWLTG
MB8GA1UdIwQYMBaAFP/veqML5yPinOtYDEERVimhWLTGMAwGA1UdEwQFMAMBAf8w
CgYIKoZIzj0EAwIDSAAwRQIhALtjU5ohBi8VwToP0Ns56Dy+P03gs8bZEF6vqT1Y
EAZRAiBkFW/6byxrFvbU4yQrPEACPZH8P1v7wOnmX26AwbZ1Tw==
-----END CERTIFICATE-----
```

We can see the certificate issuer information are different from our CA and the email used look suspect: `man.in.the.middle@hackers.com`

Verify if the certificate is issued by our CA:

```
$ openssl verify -CAfile rootCAcert.pem isaac_cert.
pem
C = ES, ST = Barcelona, L = Barcelona, O = UPC, CN = Sir Isaac Newton, emailAddress =
man.in.the.middle@hackers.com
error 18 at 0 depth lookup: self-signed certificate
error isaac_cert.pem: verification failed
```

> ✕ **Error**
>
> Isaac's certificated is not issued by our CA

4.5.2.2 Check CRL validity

CRL content:

```
$ openssl crl -in crl_002.pem -noout -text
Certificate Revocation List (CRL):
        Version 2 (0x1)
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: C = ES, ST = Barcelona, L = Barcelona, O = UPC, OU = MAK, CN = Jorge,
emailAddress = jorge.villar@upc.edu
        Last Update: Nov 27 19:01:38 2023 GMT
        Next Update: Dec 27 19:01:38 2023 GMT
        CRL extensions:
            X509v3 CRL Number:
                4
Revoked Certificates:
    Serial Number: E931CA09FE2F3A7B
        Revocation Date: Nov 16 16:37:13 2023 GMT
        CRL entry extensions:
```

```
            X509v3 CRL Reason Code:
                Key Compromise
    Serial Number: E931CA09FE2F3A7E
        Revocation Date: Nov 27 18:59:38 2023 GMT
        CRL entry extensions:
            X509v3 CRL Reason Code:
                Key Compromise
    Signature Algorithm: ecdsa-with-SHA256
    Signature Value:
        30:45:02:20:75:ca:7b:3c:01:0d:0e:24:71:f1:68:f4:8d:63:
        96:bb:92:df:50:fb:5e:3d:0c:51:e1:91:a0:05:1d:d8:54:37:
        02:21:00:e0:aa:96:66:03:5f:a7:0e:dc:6f:bb:29:61:22:94:
        a8:90:57:2f:00:47:b8:b9:9d:67:d5:78:24:7b:dc:dd:c6
```

Verify it is issued by our CA:

```
$ openssl crl -in crl_002.pem -CAfile rootCAcert.pem -noout -verify
verify OK
```

> ✓ **Valid**
>
> The CRL was issued by our CA.

4.5.2.3 Check certificate revocation

In the `crl_002.pem` file there are two revoked certificates:

```
...
Revoked Certificates:
    Serial Number: E931CA09FE2F3A7B
    ...
    Serial Number: E931CA09FE2F3A7E
...
```

Let's compare these serial numbers with the public certificate we have:

```
$ openssl x509 -in isaac_cert.pem -serial -noout
serial=EE37C49F49AB17C3
```

> ✓ **Valid**
>
> Isaac's certificate serial number is not in the revocated list!

4.5.2.4 Check signature validity

Decode the signature:

```
cat sig_002.b64 | openssl enc -d -a > sig2.bin
```

Content of `sig2.bin`:

```
$ cat sig2.bin
0E f nx K Aq 0 큅 J 'q Ù+ ! =lF%
_t T OL T P n5
```

Then verify it:

```
$ openssl pkeyutl -certin -inkey isaac_cert.pem -verify -rawin -in flat_e.pdf -sigfile
sig2.bin
Signature Verified Successfully
```

> ✓ **Valid**
>
> Signature is valid: the pdf was not altered!

4.5.3 Niels

4.5.3.1 Check signature validity

Niels' certificate:

```
$ openssl x509 -in niels_cert.pem -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            e9:31:ca:09:fe:2f:3a:7e
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: C = ES, ST = Barcelona, L = Barcelona, O = UPC, OU = MAK, CN = Jorge,
emailAddress = jorge.villar@upc.edu
        Validity
            Not Before: Nov 27 18:43:10 2023 GMT
            Not After : Nov 26 18:43:10 2024 GMT
        Subject: C = ES, ST = Barcelona, O = UPC, CN = Niels Bohr, emailAddress =
nbohr@upc.edu
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:3c:55:06:3d:28:14:22:c4:92:9d:57:2b:1d:43:
                    87:26:0d:61:6c:47:2c:39:3a:6f:4b:9b:27:59:f6:
                    6a:3a:01:98:79:77:4f:81:ea:00:99:3e:e7:67:2b:
```

```
                    3d:63:9a:65:00:8b:93:1f:fb:5f:df:f4:3c:1f:87:
                    d7:e2:46:1c:90
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                F8:BF:68:27:83:BE:5B:5A:0D:AC:8D:53:5C:9F:25:9A:7A:0B:E9:32
            X509v3 Authority Key Identifier:
                F5:3C:59:F9:A4:FA:CD:C3:0D:A2:D4:12:BA:5A:F9:41:EA:19:48:2C
    Signature Algorithm: ecdsa-with-SHA256
    Signature Value:
        30:46:02:21:00:84:b4:d4:86:38:e3:53:56:39:6e:cc:c7:b6:
        7d:0b:96:95:a8:5c:8e:db:75:fd:31:36:51:ed:59:94:b7:9b:
        a9:02:21:00:89:91:ad:71:09:0e:af:35:93:e7:d7:36:ae:61:
        e3:c8:8b:95:6a:2e:1a:8a:1d:e6:75:22:d7:4a:82:a5:d8:cc
-----BEGIN CERTIFICATE-----
MIICXDCCAgGgAwIBAgIJAOkxygn+Lzp+MAoGCCqGSM49BAMCMIGGMQswCQYDVQQG
EwJFUzESMBAGA1UECAwJQmFyY2Vsb25hMRIwEAYDVQQHDAlCYXJjZWxvbmExDDAK
BgNVBAoMA1VQQzEMMAoGA1UECwwDTUFLMQ4wDAYDVQQDDAVKb3JnZTEjMCEGCSqG
SIb3DQEJARYUam9yZ2UudmlsbGFyQHVwYy5lZHUwHhcNMjMxMTI3MTg0MzEwWhcN
MjQxMTI2MTg0MzEwWjBiMQswCQYDVQQGEwJFUzESMBAGA1UECAwJQmFyY2Vsb25h
MQwwCgYDVQQKDANVUEMxEzARBgNVBAMMCk5pZWxzIEJvaHIxHDAaBgkqhkiG9w0B
CQEWDW5ib2hyQHVwYy5lZHUwWTATBgcqhkjOPQIBBggqhkjOPQMBBwNCAAQ8VQY9
KBQixJKdVysdQ4cmDWFsRyw5Om9LmydZ9mo6AZh5d0+B6gCZPudnKz1jmmUAi5Mf
+1/f9Dwfh9fiRhyQo3sweTAJBgNVHRMEAjAAMCwGCWCGSAGG+EIBDQQfFh1PcGVu
U1NMIEdlbmVyYXRlZCBDZXJ0aWZpY2F0ZTAdBgNVHQ4EFgQU+L9oJ4O+W1oNrI1T
XJ8lmnoL6TIwHwYDVR0jBBgwFoAU9TxZ+aT6zcMNotQSulr5QeoZSCwwCgYIKoZI
zj0EAwIDSQAwRgIhAIS01IY441NWOW7Mx7Z9C5aVqFyO23X9MTZR7VmUt5upAiEA
iZGtcQkOrzWT59c2rmHjyIuVai4aih3mdSLXSoKl2Mw=
-----END CERTIFICATE-----
```

Verify if the certificate is issued by our CA:

```
$ openssl verify -CAfile rootCAcert.pem niels_cert.pem
niels_cert.pem: OK
```

> ✓ **Valid**
>
> The certificate was issued by our CA!

4.5.3.2 Check CRL validity

CRL content:

```
$ openssl crl -in crl_002b.pem -noout -text
Certificate Revocation List (CRL):
        Version 2 (0x1)
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: C = ES, ST = Barcelona, L = Barcelona, O = UPC, OU = MAK, CN = Jorge,
emailAddress = jorge.villar@upc.edu
        Last Update: Nov 27 19:01:38 2023 GMT
        Next Update: Dec 27 19:01:38 2023 GMT
        CRL extensions:
            X509v3 CRL Number:
                4
Revoked Certificates:
    Serial Number: E931CA09FE2F3A7B
        Revocation Date: Nov 16 16:37:13 2023 GMT
        CRL entry extensions:
            X509v3 CRL Reason Code:
                Key Compromise
    Serial Number: E931CA09FE2F3A82
        Revocation Date: Nov 27 18:59:38 2023 GMT
        CRL entry extensions:
            X509v3 CRL Reason Code:
                Key Compromise
    Signature Algorithm: ecdsa-with-SHA256
    Signature Value:
        30:45:02:20:75:ca:7b:3c:01:0d:0e:24:71:f1:68:f4:8d:63:
        96:bb:92:df:50:fb:5e:3d:0c:51:e1:91:a0:05:1d:d8:54:37:
        02:21:00:e0:aa:96:66:03:5f:a7:0e:dc:6f:bb:29:61:22:94:
        a8:90:57:2f:00:47:b8:b9:9d:67:d5:78:24:7b:dc:dd:c6
```

Verify it is issued by our CA:

```
$ openssl crl -in crl_002b.pem -CAfile rootCAcert.pem -noout -verify
verify failure
```

✕ Error

The CRL was NOT issued by our CA.

If we repeat the check using the file `crl_003.pem` in which we have trust:

```
$ openssl crl -in crl_003.pem -CAfile rootCAcert.pem -noout -verify
verify OK
```

If we compare their content we can see some little differences:

```
$ openssl crl -in crl_003.pem -noout -text
Certificate Revocation List (CRL):
```

```
        Version 2 (0x1)
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: C = ES, ST = Barcelona, L = Barcelona, O = UPC, OU = MAK, CN = Jorge,
emailAddress = jorge.villar@upc.edu
        Last Update: Nov 28 11:27:54 2023 GMT
        Next Update: Dec 28 11:27:54 2023 GMT
        CRL extensions:
            X509v3 CRL Number:
                5
Revoked Certificates:
    Serial Number: E931CA09FE2F3A7B
        Revocation Date: Nov 16 16:37:13 2023 GMT
        CRL entry extensions:
            X509v3 CRL Reason Code:
                Key Compromise
    Serial Number: E931CA09FE2F3A7E
        Revocation Date: Nov 27 18:59:38 2023 GMT
        CRL entry extensions:
            X509v3 CRL Reason Code:
                Key Compromise
    Signature Algorithm: ecdsa-with-SHA256
    Signature Value:
        30:46:02:21:00:e4:61:de:aa:03:74:1f:9a:29:d7:c3:d2:21:
        1f:10:df:70:b4:f4:86:53:04:7d:78:49:59:9b:07:e4:0e:46:
        ae:02:21:00:e8:b4:16:47:9c:87:89:78:dc:40:4d:bf:17:6e:
        69:df:0a:12:db:4e:95:ab:e0:8d:a4:b1:84:4a:4a:a3:19:fc
```

The differences are:

1. Last Update
   - `crl_003.pem`: Nov 28 11:27:54 2023 GMT
   - `crl_002b.pem`: Nov 27 19:01:38 2023 GMT
2. Next Update
   - `crl_003.pem`: Dec 28 11:27:54 2023 GMT
   - `crl_002b.pem`: Dec 27 19:01:38 2023 GMT
3. CRL Number
   - `crl_003.pem`: X509v3 CRL Number is 5.
   - `crl_002b.pem`: X509v3 CRL Number is 4.
4. Revoked Certificates
   - Both files have a certificate with Serial Number E931CA09FE2F3A7B revoked on Nov 16 16:37:13 2023 GMT for Key Compromise.
   - `crl_003.pem` also has a certificate with Serial Number E931CA09FE2F3A7E revoked on Nov 27 18:59:38 2023 GMT for Key Compromise.
   - `crl_002b.pem` has a different certificate with Serial Number E931CA09FE2F3A82 revoked on Nov 27 18:59:38 2023 GMT for Key Compromise.
5. Signature Value
   - The signature values in both files are different, indicating that the data in the CRL and/or the signing process was not identical.

We can assume the file `crl_002b.pem` was corrupted.

4.5.3.3 Check certificate revocation

In the `crl_002b.pem` there are two revoked certificates:

```
...
Revoked Certificates:
    Serial Number: E931CA09FE2F3A7B
    ...
    Serial Number: E931CA09FE2F3A82
...
```

Let's compare these serial numbers with the public certificate we have:

```
$ openssl x509 -in niels_cert.pem -serial -noout
serial=E931CA09FE2F3A7E
```

> ✓ **Valid**
>
> Niels' certificate serial number is not in the revocated list!

Now that we have a doubt with the `crl_002b.pem` file, let's repeat the check with our trusted `crl_003.pem` file:

```
Revoked Certificates:
    Serial Number: E931CA09FE2F3A7B
    ...
    Serial Number: E931CA09FE2F3A7E
```

Let's compare these serial numbers with the public certificate we have:

```
$ openssl x509 -in niels_cert.pem -serial -noout
serial=E931CA09FE2F3A7E
```

> × **Error**
>
> Niels' certificate serial number is revocated!

We can assure the CRL file was modified to "un-revoke" Niels' certificate and impersonate him

4.5.3.4 Check signature validity

Decode the signature:

```
cat sig_002.b64 | openssl enc -d -a > sig2.bin
```

Content of `sig2.bin`:

```
$ cat sig3.bin
0E Z  Q  *#] Yr )
7  .  m  ?ç !  Ob /  i,  5 f  w3  aW  f
```

Then verify it:

```
$ openssl pkeyutl -certin -inkey niels_cert.pem -verify -rawin -in g_dice.pdf -sigfile
sig3.bin
Signature Verified Successfully
```

> ✓ **Valid**
>
> Signature is valid: the pdf was not altered!

Even if the signature looks valid, because the authentication chain was broken we can't assume this message was indeed signed by Niels.

4.5.4 Conclusion

Checks 1 to 3 could also be done with the following one-liner:

```
$ openssl verify -CAfile rootCAcert.pem -check_ss_sig -CRLfile crl_00X.pem -crl_check
peer_cert.pem
```

Albert:

- [x] Certificate authenticity
- [x] CRL validity
- [x] Certificate revocation
- [x] Signature validity

Isaac:

- [ ] Certificate authenticity
- [x] CRL validity
- [x] Certificate revocation
- [x] Signature validity

Niels:

- [x] Certificate authenticity

- [ ] CRL validity
- [ ] Certificate revocation
- [ ] Signature validity

4.6 Send report digitally signed

4.6.1 Sign report

Since we have a public certificate from the CA we can use our previously generated private key to sign the document.

We sign our report and save the signature in `sign.bin`:

```
openssl pkeyutl -inkey myECpkey.pem -sign -rawin -in lab4_report.pdf -out sign.bin
```

Convert bytes to `base64`:

```
cat sign.bin | base64 > sigReport.b64
```

Show the signature:

```
$ cat sigReport.b64
MEUCIFbX5Mj2Bb7eKIywQ5bxE7Ina3+Eog+Hx/TquAxjMuO+AiEAms+nlmIeYB/siGcPJKcJQXJf
EP36lOIBq03W/iiQ2Ts=
```

> ✓ **Success**
>
> The report can now be send with the public certificate and the signature

4.6.2 Verify the signature

To verify our signature we first need to decode the signature:

```
cat sigReport.b64 | openssl enc -d -a > sigReport.bin
```

Then use our public certificate to verify it:

```
$ openssl pkeyutl -certin -inkey 102_cert.pem -verify -rawin -in lab4_report.pdf -
sigfile sigReport.bin
Signature Verified Successfully
```

> ✓ **Success**

Lab done!