

Universidade do Minho

MESTRADO EM ENGENHARIA INFORMÁTICA

**Engenharia de segurança**

**Ficha de exercício 5**

**Grupo 1**

RUI CARLOS AZEVEDO CARVALHO - PG47633

DANIEL BARBOSA MIRANDA - PG47123

ANA LUÍSA LIRA TOMÉ CARNEIRO - PG46983

## Parte VIII: Infraestrutura de chave pública

### Pergunta P1.1

#### Exercício 1

A pergunta 1 consistia em identificar e analisar as várias componentes de um certificado de chave móvel digital, *CertExchange.cer*. Para isso utilizou-se o software OpenSSL que permitiu analisar o conteúdo do certificado a partir do comando:

```
openssl x509 -inform DER -in CertExchange.cer -text -noout
```

Tal como podemos ver pela a figura 1 o certificado da chave móvel digital é composto por:

- **Version:** Corresponde a versão standard do X509. Os Certificados X509 corresponde ao tipo de certificados de chave pública que possuem uma estrutura standard entre eles e foram introduzidos para autenticar nós do serviço de diretoria X.500.
- **Serial Number:** Número único atribuído pela Entidade de Certificação (EC), isto é, a entidade responsável por emitir, renovar ou revogar certificados. Este número único visa identificar o certificado que estamos a analisar.
- **Signature Algorithm:** Este parâmetro identifica o algoritmo usado para gerar a assinatura digital associada a este certificado. Tal como podemos ver o algoritmo utilizado é o SHA256 com uma cifra RSA.
- **Issuer:** Corresponde à identidade, EC, que emitiu o certificado. Esta componente tem uma estrutura hierárquica estandardizada pelas normas X.520.
  - C : país de emissão que neste caso será Portugal;
  - O : organização de emissão do certificado. Como este certificado é um certificado de chave móvel digital associado ao cartão de cidadão então a organização de emissão será o Cartão de Cidadão;
  - OU : Corresponde à unidade organizacional dentro da organização que neste caso está associada à assinatura digital a partir da chave móvel digital. Pode haver mais do que uma unidade organizacional, tal como podemos ver pela a imagem.
- **Validity:** Apresenta a validade do certificado. É de notar que esta validade está associada à data de emissão e registo do certificado (*not before*) e à data de validade do cartão de cidadão (*not after*), ou seja, quando o cartão de cidadão perde a validade, o certificado também perde.
- **Subject:** De forma semelhante à componente *Issuer* também o *Subject* apresenta uma estrutura hierárquica que identifica o titular da chave pública contida no certificado. Para além dos parâmetros já apresentados na componente *Issuer*, esta componente apresenta ainda os seguintes parâmetros.

- SN: Representa o *serial number* associado ao titular, neste caso é os apelidos do titular da chave pública;
- GN : Representa o *given name* associado ao titular, isto é, o nome próprio do titular do certificado.
- serialNumber : O *serial number* do certificado.
- CN: Representa o *common name* associado ao titular, isto é, o nome completo do titular do certificado.

```
Certificate:
Data:
  Version: 3 (0x2)
  Serial Number:
    67:7a:7b:11:67:ff:06:ca
  Signature Algorithm: sha256WithRSAEncryption
  Issuer: C = PT, O = AMA - AG\C3\8ANCIA PARA A MODERNIZA\C3\87\C3\830 ADMINISTRATIVA I. P., OU = Cart\C3\A3o de Ciudad\C3\A3o, OU = subECEstado, CN = EC de Chave M\C3\B3vel Digital de Assinatura Digital Qualificada do Cart\C3\A3o de Ciudad\C3\A3o 00
004
  Validity
    Not Before: Apr  7 15:59:35 2022 GMT
    Not After : Mar 22 23:59:59 2025 GMT
  Subject: C = PT, O = Cart\C3\A3o de Ciudad\C3\A3o, OU = Chave M\C3\B3vel Digital de Assinatura Qualificada do Ciudad\C3\A3o, OU = Ciudad\C3\A3o, OU = RemoteQSCDManagement, SN = LIRA TOM\C3\89 CARNEIRO, GN = ANA LU\C3\8DSA, serialNumber = BI14691196,
  CN = ANA LU\C3\8DSA LIRA TOM\C3\89 CARNEIRO
```

Figure 1: Componentes do certificado CMD - Componentes base

Além do demonstrado na figura 1, podemos ver pela figura 2 que o certificado da chave móvel digital é ainda composto pela estrutura **Subject Public Key Info** que contém a chave pública de 3072 bits do titular do certificado (**RSA Public-Key**) e identificação do algoritmo correspondente (**Public Key Algorithm**), que neste caso é a cifra RSA.

```
Subject Public Key Info:
  Public Key Algorithm: rsaEncryption
  RSA Public-Key: (3072 bit)
  Modulus:
    00:88:f4:8f:5f:50:0b:eb:cc:2e:c0:ad:de:da:05:
    20:78:d1:6b:5e:b7:d3:da:f7:e0:30:31:07:c9:8c:
    7b:6d:96:c4:79:5b:c1:92:a4:bb:38:30:36:a6:7b:
    84:71:e6:6b:12:32:8e:55:26:00:28:3d:5e:30:05:
    8f:25:d8:ef:fa:d2:f5:4e:57:20:f4:b1:ce:92:3f:
    52:d6:82:c8:24:dc:c0:39:46:81:cf:f2:5c:00:1f:
    e8:16:42:7f:e8:f5:45:50:43:36:60:f5:e2:02:69:
    e4:6b:b9:ca:82:0d:5b:67:e7:cc:33:b8:f8:4d:ba:
    d9:44:08:15:7b:e3:46:ed:5d:92:58:17:73:16:25:
    3a:c7:58:0f:3e:3e:38:5b:25:2c:46:8e:8d:d2:d2:
    2f:6e:c1:f4:63:42:35:ab:8c:a3:35:5a:9b:5d:e3:
    1a:37:59:74:b3:c6:b9:3c:50:d9:33:71:6c:54:52:
    81:ca:b9:71:c0:15:55:d4:78:23:3c:e6:01:4c:17:
    d4:f4:f4:58:05:7f:21:c6:a6:77:91:8e:a0:e0:4b:
    d6:ae:33:35:c4:84:df:6f:91:c7:44:70:38:4e:ee:
    17:bc:d1:31:48:70:cc:e0:f8:67:6b:73:2c:92:54:
    66:5b:d1:4b:c6:a6:f5:13:31:60:94:3b:8a:f1:98:
    03:a0:73:cf:b5:1b:c8:7c:7c:05:e9:97:a2:27:6f:
    1b:ef:1d:ce:a0:d5:36:76:34:a0:9e:6f:3e:50:a0:
    9e:76:54:cc:69:cb:12:0b:6b:bd:47:38:bb:58:b0:
    30:2f:a0:21:fd:df:e5:39:1d:1e:f1:35:45:d9:0b:
    4a:e4:53:45:73:a0:76:90:89:ee:3c:96:30:0a:0c:
    1f:b8:5c:c6:6b:ca:a7:ee:ae:9b:90:50:9a:0d:08:
    83:3a:14:a4:91:c0:4c:60:b3:c1:c4:78:17:b4:28:
    1e:32:ba:3a:f6:e8:2f:c2:78:b7:ec:95:f6:74:e0:
    e6:10:ec:e0:a5:c7:3f:08:7b:e9
  Exponent: 65537 (0x10001)
```

Figure 2: Componentes do certificado CMD - Informação sobre chave pública

Adicionalmente às componentes standard apresentadas, o standard X.509 também define várias extensões destinadas a descreverem como é que o certificado podem ser utilizado, apresentado um conjunto variado de atributos que limitando e esclarecem o uso do certificado. Neste certificado podemos ver as seguintes extensões:

- **Basic Constraints:** Extensão utilizada nos certificados CA. Tal como podemos ver na figura 3, como este não se trata de um certificado CA então a extensão não é aplicada;
- **Authority Key Identifier:** Extensão que fornece meios de identificar a chave pública correspondente à chave privada usada para assinar o certificado;
- **Certificate Policies:** Esta extensão permite identificar as políticas e informação sobre o funcionamento do certificado;
- **Subject Directory Attributes:** Contém uma sequência de elementos, onde cada elemento descreve um atributo não estandardizado que se deseja adicionar ao certificado;
- **CRL Distribution Points:** Identifica como obter a lista com a informação sobre a revogação de certificados;
- **Subject Key Identifier:** Fornece um meio de identificar certificados que contêm uma chave pública específica;
- **Key Usage:** Define o propósito da chave do certificado.

```
X509v3 extensions:
X509v3 Basic Constraints: critical
CA:FALSE
X509v3 Authority Key Identifier:
keyid:A1:03:A5:82:86:C3:E1:7E:32:F7:88:F0:8E:52:68:D1:0D:9C:24:AC

Authority Information Access:
OCSP - URI:http://ocsp.cmd.cartaodecidadao.pt/publico/ocsp

X509v3 Freshest CRL:

Full Name:
URI:http://pki.cartaodecidadao.pt/publico/lrc/cc_sub-ec_cidadao_cmd_crl0004_delta_p0001.crl

X509v3 Certificate Policies:
Policy: 2.16.620.1.1.1.2.10
CPS: http://www.scee.gov.pt/pcert
Policy: 0.4.0.194112.1.2
Policy: 2.16.620.1.1.1.2.4.3.0.1.1
CPS: http://pki.cartaodecidadao.pt/publico/politicas/pc/cc_sub-ec_cidadao_cmd_pc.html
Policy: 2.16.620.1.1.1.2.4.3.0.7
CPS: http://pki.cartaodecidadao.pt/publico/politicas/dpc/cc_sub-ec_cidadao_cmd_dpc.html
Policy: 0.4.0.2042.1.2

X509v3 Subject Directory Attributes:
0.0...+.....1...20000221120000Z
qcStatements:
0.0.....F..0.....F..0.....F..0.....F..0..0?9https://pki.cartaodecidadao.pt/publico/politicas/cps.html..PT0?9https://pki.cartaodecidadao.pt/publico/politicas/cps.html..EN
X509v3 CRL Distribution Points:

Full Name:
URI:http://pki.cartaodecidadao.pt/publico/lrc/cc_sub-ec_cidadao_cmd_crl0004_p0001.crl

X509v3 Subject Key Identifier:
01:55:3F:73:FA:2E:FE:14:EC:2B:B4:DF:8F:65:D1:19:5C:2D:EC:DE
X509v3 Key Usage: critical
Non Repudiation
```

Figure 3: Componentes do certificado CMD - Extensões

Finalmente, é apresentado na secção **Signature Algorithm** o algoritmo utilizado para a geração da chave pública tal como já foi mencionado e a chave pública associada ao certificado.

```
Signature Algorithm: sha256WithRSAEncryption
a5:1f:41:79:96:f8:a0:22:73:84:c9:17:6a:48:dc:b1:13:d0:
7e:2b:65:8f:8a:ff:84:00:70:c7:01:e4:c0:6f:4c:4b:d2:a0:
b5:fc:ae:d6:2a:0b:21:06:38:2b:1e:1b:27:d2:bd:4f:74:fb:
48:66:6e:8a:eb:50:a1:e9:ec:bb:bf:7b:c9:16:cd:1a:5a:27:
19:81:94:55:c5:9e:e6:b1:32:b8:14:7f:66:44:50:2f:7e:77:
b2:1d:db:d0:d7:94:97:85:d4:8e:6f:d7:b7:36:9f:ae:82:44:
0f:6a:b4:1f:8b:03:ac:1b:bc:9e:e2:79:2c:d9:d4:e6:66:98:
ca:7c:5c:04:62:51:7c:ec:61:9c:44:e4:56:33:1e:cd:7e:04:
f5:53:5b:1d:60:7e:51:33:5d:c4:ac:74:9f:38:4b:2e:b5:0f:
41:c4:ae:be:c8:ab:97:b5:65:3d:bd:94:be:72:fb:cf:bd:23:
c7:5f:56:e2:be:04:b6:e5:1d:87:1d:87:3c:f4:59:48:fc:4c:
10:89:4a:34:34:4f:73:d1:54:7e:2a:2a:dc:95:e2:68:94:2d:
01:83:eb:5b:8b:a7:d1:b5:da:fd:c6:29:c4:8f:3e:5b:91:e9:
9f:e3:4c:f6:82:eb:74:80:6f:3c:53:14:43:e2:e6:89:38:7e:
3d:82:23:df:49:e3:67:d3:c6:21:6e:fd:b2:b8:7c:94:fb:6d:
12:ca:58:cf:db:1e:27:7c:3e:a2:5a:5b:a2:31:75:1f:0a:c6:
14:de:74:47:d3:77:d0:ae:ac:15:f0:82:b7:29:0f:87:6a:57:
2e:b7:39:48:75:72:ec:e4:73:32:93:14:c9:29:b2:00:fe:1b:
f9:29:43:e1:a0:55:f1:61:bf:b5:6d:f8:a5:48:67:eb:d2:bb:
d3:df:b1:ec:25:47:94:3e:57:82:06:c5:c9:f6:4f:df:61:91:
80:4b:cf:a2:79:32:cb:7f:55:ff:13:76:d8:c8:88:23:b5:38:
7c:b2:81:e0:ea:1f:63:87:58:af:bb:72:07:ea:df:c4:9e:8e:
39:ac:ee:87:5f:6b:1c:56:54:78:6d:37:e1:90:58:19:06:c4:
17:67:95:be:b1:00:c8:e3:11:52:7e:35:e2:a7:c5:f0:c7:0a:
a5:3f:61:86:0f:8f:f7:cc:a3:cd:82:56:9d:ae:fa:4d:70:94:
22:91:81:97:12:cc:c5:5a:81:a1:e2:33:77:6f:2c:32:b8:8e:
7c:39:82:a1:0f:b5:af:4e:7f:9b:d5:67:fb:0a:65:4e:f5:b6:
b7:a5:ee:8d:d0:d3:8a:64:61:b9:ee:81:6a:e7:33:1c:9e:41:
ee:d9:d7:55:9f:b9:ca:2a
```

Figure 4: Componentes do certificado CMD - Chave Pública

## Exercício 2

As bibliotecas principais que foram utilizadas foram as seguintes:

```
from cryptography import x509
from cryptography.x509.oid import ExtensionOID
import requests
```

De facto, a biblioteca "Cryptography" permite a utilização de funções que permitem carregar bytes que correspondem a certificados para objetos x509, sendo que, é possível utilizar sobre estes métodos que permitem obter informações acerca do certificado correpondente, tal como o *issuer* ou até mesmo as extensões. Para além disto a biblioteca requests permite realizar pedidos HTTP, sendo útil para ir buscar o ficheiro CRL cujo URL se encontra no certificado do cartão de cidadão.

Inicialmente é necessário obter o URL do ficheiro CRL, logo para isto é construído um objeto x509 através do método "load\_der\_x509\_certificate" ou "load\_pem\_x509\_certificate" dependendo da extensão do ficheiro do certificado:

```
1 certificateName = sys.argv[1]
2 crlFileName = sys.argv[2]
3
4 f = open(certificateName, 'rb')
5
6 if certificateName.endswith(".cer"):
```

```

7     cert = x509.load_der_x509_certificate(f.read())
8
9     else:
10        cert = x509.load_pem_x509_certificate(f.read())

```

De seguida a função "getCRLfile" começa por obter as extensões do certificado e obtém a que corresponde ao local onde está o URL do ficheiro CRL, sendo que, de seguida imprime esse URL no ecrã e faz um pedido HTTP para o obter e guarda-o num ficheiro cujo nome é fornecido pelo utilizador:

```

1  def getCRLfile():
2      crl =
3          ↪ cert.extensions.get_extension_for_oid(ExtensionOID.CRL_DISTRIBUTION_POINTS)
4      crl_url = crl.value[0].full_name[0].value
5
6      print("CRL URL: " + str(crl_url))
7
8      crl_file = requests.get(crl_url)
9
10     p = open(crlFileName, 'wb')
11     p.write(crl_file.content)
12     p.close()

```

Por fim a função check carrega este ficheiro CRL para um objeto através do método "load\_der\_x509\_crl" e verifica se o *serial\_number* presente no certificado original existe na lista dos *serial numbers* que foram revogados. Caso o certificado tenha sido revogado, então é possível utilizar o método "revocation\_date" para obter a data onde este foi revogado. Por conseguinte utiliza também os métodos "last\_update" e "next\_update" para obter as datas do último update da CRL atual e em que data é que vai ser efetuada o próximo update.

```

1  def check():
2      f = open(crlFileName, 'rb')
3
4      crl = x509.load_der_x509_crl(f.read())
5      revoked =
6          ↪ crl.get_revoked_certificate_by_serial_number(cert.serial_number)
7
8      print("Data da CRL atual: " + str(crl.last_update))
9      print("Data da próxima CRL: " + str(crl.next_update))
10
11     if revoked == None:
12         print("O certificado não foi revogado!")
13
14     else:
15         print("O certificado foi revogado!")
16         print("Data: " + str(revoked.revocation_date))

```

Para utilizar este programa (que se encontra neste [link](#)) basta utilizar o seguinte comando:

```
python3 ex2.py [CERTIFICATE PATH] [CRL FILE NAME]
```

O resultado de aplicar este comando ao certificado utilizado no exercício anterior é o seguinte:

```
CRL URL: http://pki.cartaodecidadao.pt/publico/lrc/cc_sub-ec_cidadao_cmd_crl0004_p0001.crl
Data da CRL atual: 2022-04-14 19:01:42
Data da próxima CRL: 2022-04-21 19:01:42
0 certificado não foi revocado!
```

Figure 5: Verificação de revogação do certificado utilizado no exercício 1

### Exercício 3

O objetivo deste exercício é o desenvolvimento de um programa capaz de receber um certificado Cartão de Cidadão ou um certificado Chave Móvel Digital e indicar o estatuto do mesmo relativamente à sua revogação utilizando para tal OCSP.

```
1 import os, sys, subprocess
2 sys.tracebacklimit = 0
```

Assim as bibliotecas importadas para este trabalho são `os` que servirá para reiniciar o programa em caso de erro de certificado e para remover certos ficheiros no final da sua execução.

Quanto à biblioteca `sys` apenas serve para remover os tracebacks e manter a *bash* mais limpa, nos casos de erro.

A biblioteca `subprocess` servirá para executar comandos na *bash* essenciais às diferentes fases de implementação do exercício, sendo estes comandos baseados nos *slides* das aulas.

```
1 if __name__ == "__main__":
2     print("\nNome do certificado: ", end = '')
3     certificate = input()
4     status = ex3(certificate)
5     if status != None:
6         print(status)
```

Ao iniciar o programa será pedido o nome do ficheiro do certificado, sendo que depois efetua-se a chamada da função propriamente dita do programa em questão, que caso retorne `None` o programa simplesmente termina.

É relevante mencionar que o programa apenas aceita ficheiros de certificado do tipo `.cer` ou `.pem`.

```
1 def ex3(certificate):
2
3     final = None
4
5     # se o ficheiro for do tipo .cer converter para pem
6     if certificate.endswith(".cer"):
7
```

```

8 converter = "openssl x509 -inform Der -in " + certificate + "
  ↪ -out CertExchange.pem"
9 subprocess.call(converter, shell=True)
10 certificate = "CertExchange.pem"

```

Como se pode observar a função recebe o certificado, começando declarar o que será o valor de retorno, que por *default* é `None`.

De seguida verifica-se se o tipo do certificado recebido é `.cer` e se o for, é gerado, com comandos `openssl`, um ficheiro correspondente à sua conversão para PEM.

```

1 if certificate.endswith(".pem"):
2
3     # obter linha do issuer
4     get_issuer = "openssl x509 -in " + certificate + " -noout
  ↪ -issuer"
5     issuer_name = str(subprocess.check_output(get_issuer,
  ↪ shell=True), 'utf-8').rstrip()

```

Nesta etapa verifica-se se o tipo do certificado é `.pem` se o certificado recebido for do tipo `.cer` este é convertido para `.pem`, chegando a esta fase desta forma e obrigatoriamente entrando na condicional.

Dentro desta o primeiro passo consiste em obter a linha do certificado que identifica o emissor do mesmo. Para tal utiliza-se comandos `openssl`, com a biblioteca `subprocess` mencionada.

```

1 # obter numero do certificado do EC
2 issuer_num = issuer_name[issuer_name.rfind(' ') + 1:]

```

De seguida, com essa linha adquire-se o número identificador da versão do certificado.

```

1 # obter certificado do EC
2 issuer_link =
  ↪ "http://pki.cartaodecidadao.pt/publico/certificado/cc_ec_cidadao_cmd/EC/%20de%"
  ↪ + issuer_num + ".cer"
3 get_issuer_link = "wget -q -O issuer.cer " + issuer_link
4 subprocess.call(get_issuer_link, shell=True)

```

Agora, é necessário adquirir o próprio certificado através do website <http://pki.cartaodecidadao.pt/>, sendo utilizado para tal, o número identificador, que permite a transferência do mesmo, com recurso ao comando `wget`, cujas flags `-q` e `-O` apenas servem, respetivamente, para omitir outputs na *bash* e atribuir o nome `issuer.cer` ao certificado.

```

1 # converter certificado do EC para pem
2 issuer_converter = "openssl x509 -inform Der -in issuer.cer -out
  ↪ issuer.pem"
3 subprocess.call(issuer_converter, shell=True)
4 issuer = "issuer.pem"

```



O certificado adquirido encontra-se no formato `.cer` e, à semelhança do caso anteriormente demonstrado, geramos o ficheiro `.pem` correspondente.

```
1 # url do serviço ocsf
2 get_url = "openssl x509 -noout -ocsp_uri -in " + certificate
3 url = str(subprocess.check_output(get_url, shell=True),
    ↪ 'utf-8').rstrip()
4 print("\nUrl do OCSF: " + url + "\n")
```

Esta etapa é utilizada para obter o *URL* do serviço de OCSF, sendo que tal pode ser feito com um comando `openssl` que o adquire da estrutura do certificado que se deseja averiguar o estatuto de revogação.

```
1 # comando de informação de revogação
2 date = "openssl ocsf -issuer " + issuer + " -cert " + certificate + "
    ↪ -url " + url + " -noverify"
3 final = str(subprocess.check_output(date, shell=True), 'utf-8')
```

Agora já se possui tudo o necessário para averiguar o estatuto em si, executa-se o comando `openssl` que requer o certificado do emissor EC, o certificado *input* e o *URL* recentemente obtido. Este comando irá retornar o estatuto de revogação do certificado em questão.

```
1 # remover ficheiros depois da execucao
2 os.remove(issuer)
3 os.remove("issuer.cer")
```

Este excerto serve apenas para remover os certificados emissores EC gerados durante o programa.

```
1 else:
2     print("\nCertificado inválido!")
3     os.system("python3 ex3.py")
4     exit()
5
6 return final
```

Esta situação corresponde a caso o certificado fornecido não fosse do tipo `.cer` ou `.pem`, seria apresentada na *bash* uma mensagem a indicar tal situação e o programa seria reiniciado. Por fim, o estatuto de revogação seria retornado.

```

rhezzus@RhEzZuS:~/Desktop/Grupo1/Pratica 1/TP5/Ex3$ python3 ex3.py
Nome do certificado: CertExchange.pem
Url do OCSP: http://ocsp.cmd.cartaodecidadao.pt/publico/ocsp
CertExchange.pem: good
This Update: Apr 12 20:17:03 2022 GMT

rhezzus@RhEzZuS:~/Desktop/Grupo1/Pratica 1/TP5/Ex3$ python3 ex3.py
Nome do certificado: CertExchange.cer
Url do OCSP: http://ocsp.cmd.cartaodecidadao.pt/publico/ocsp
CertExchange.pem: good
This Update: Apr 12 20:17:11 2022 GMT

rhezzus@RhEzZuS:~/Desktop/Grupo1/Pratica 1/TP5/Ex3$ █

```

Figure 6: Obtenção de informação de revogação do serviço OCSP

A figura acima corresponde a um exemplo efetuado com o certificado de um dos elementos do grupo. Para correr o programa basta executar `python ex3.py` na diretoria do mesmo, e fornecer como input o nome do certificado.

Como output o programa apresenta o *URL* do OCSP, o estatuto do certificado e a data de resposta do servidor.

É pertinente mencionar que, de acordo com os comandos `openssl` e os exemplos dos slides, se o estatuto do certificado fosse "revogado" seriam apresentados os mesmos dados que no exemplo, só que com dois campos adicionais, o primeiro correspondente à razão de revogação e o segundo à data de revogação.