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.

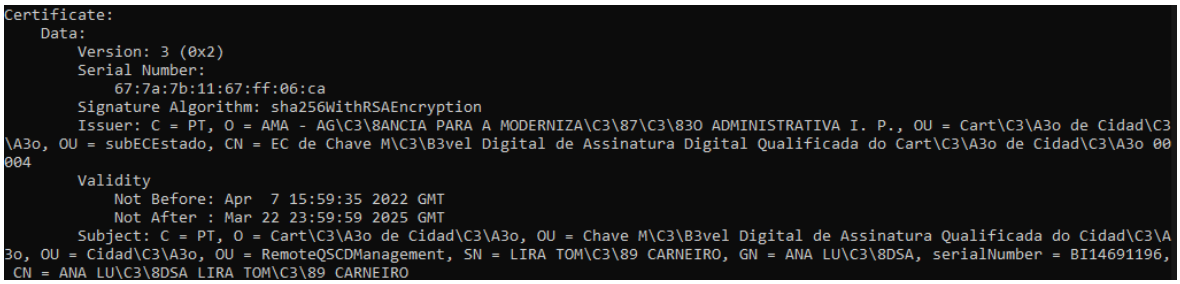

Figure 1: Componentes do certificado CMD - Componentes base

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.

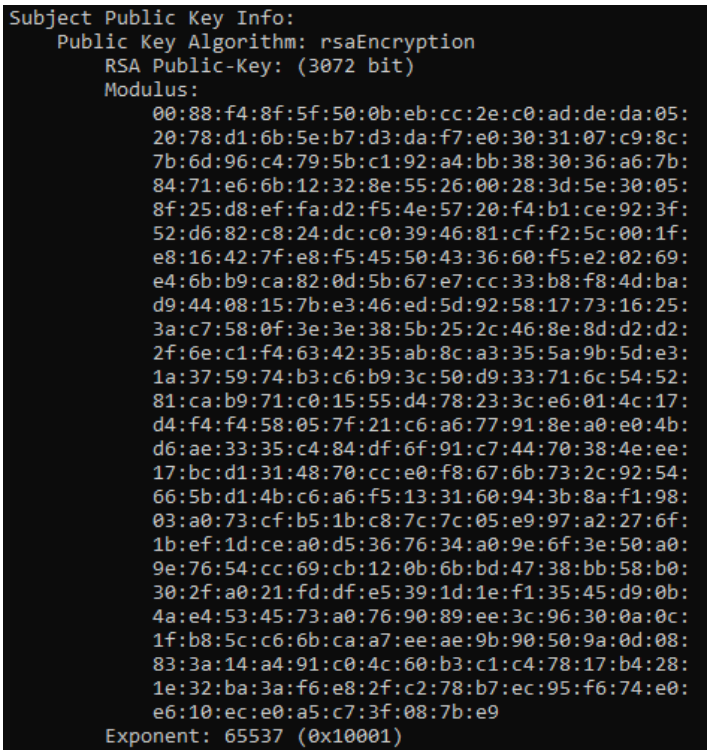

Figure 2: Componentes do certificado CMD - Informação sobre chave pública

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 Contrains**: 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.

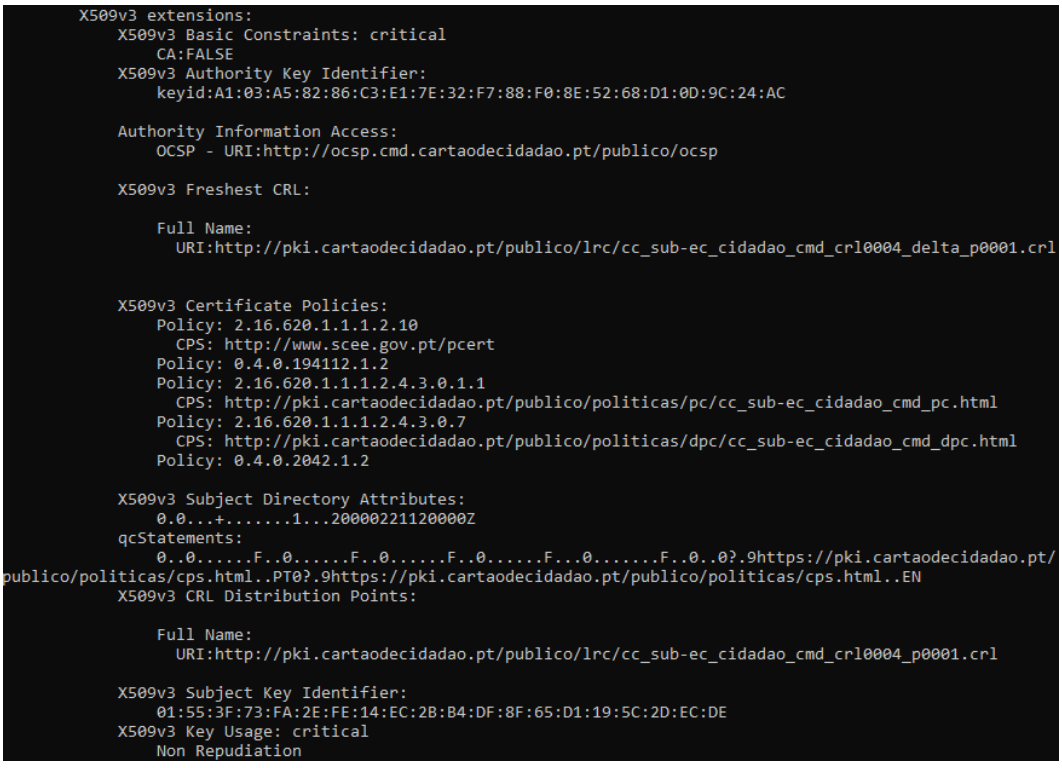

Figure 3: Componentes do certificado CMD - Extensões

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.

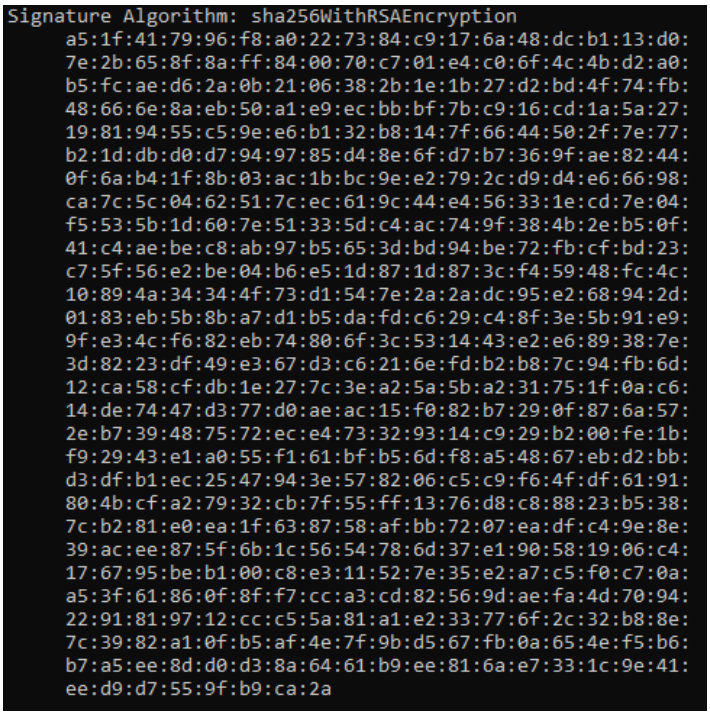

Figure 4: Componentes do certificado CMD - Chave Pública

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"):
```

```
    cert = x509.load_der_x509_certificate(f.read())

else:
    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:

```
def getCRLfile():
    crl =
    ↪  cert.extensions.get_extension_for_oid(ExtensionOID.CRL_DISTRIBUTION_POINTS)
    crl_url = crl.value[0].full_name[0].value

    print("CRL URL: " + str(crl_url))

    crl_file = requests.get(crl_url)

    p = open(crlFileName, 'wb')
    p.write(crl_file.content)
    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 orginal 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.

```
def check():
    f = open(crlFileName, 'rb')

    crl = x509.load_der_x509_crl(f.read())
    revoked =
    ↪  crl.get_revoked_certificate_by_serial_number(cert.serial_number)

    print("Data da CRL atual: " + str(crl.last_update))
    print("Data da próxima CRL: " + str(crl.next_update))

    if revoked == None:
        print("O certificado não foi revocado!")

    else:
        print("O certificado foi revogado!")
        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
O 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 pare 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 aceito 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.

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

Esta etapa é utilizada para obter o *URL* do serviço de OCSP, 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.

```
# comando de informação de revogação
date = "openssl ocsp -issuer " + issuer + " -cert " + certificate + "
↪  -url " + url + " -noverify"
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.

```
        # remover ficheiros depois da execuçao
        os.remove(issuer)
        os.remove("issuer.cer")
```

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

```
    else:
        print("\nCertificado inválido!")
        os.system("python3 ex3.py")
        exit()

    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 revocação seria retornado.

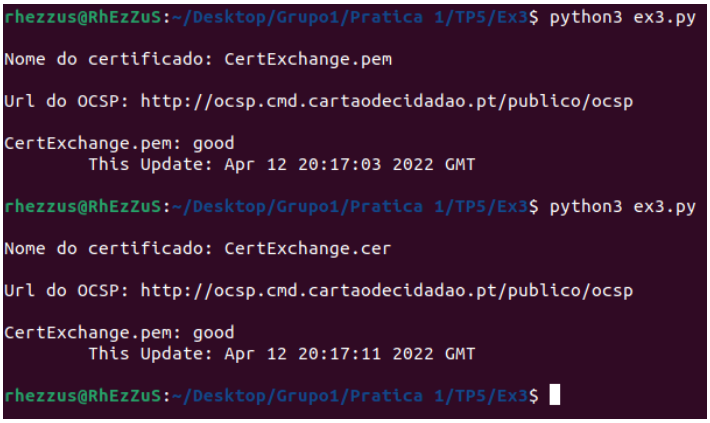

Figure 6: Obtenção de informação de revogação do serviço OCSP

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 adicionals, o primeiro correspondente à razão de revogação e o segundo à data de revogação.