# Engenharia de Segurança

16 de Março de 2021

**Grupo 7**

| | |
|---|---|
| a83899 | André Morais |
| a84485 | Tiago Magalhães |

*Prática 1 - Aula 02*

Mestrado Integrado em Engenharia Informática
Universidade do Minho

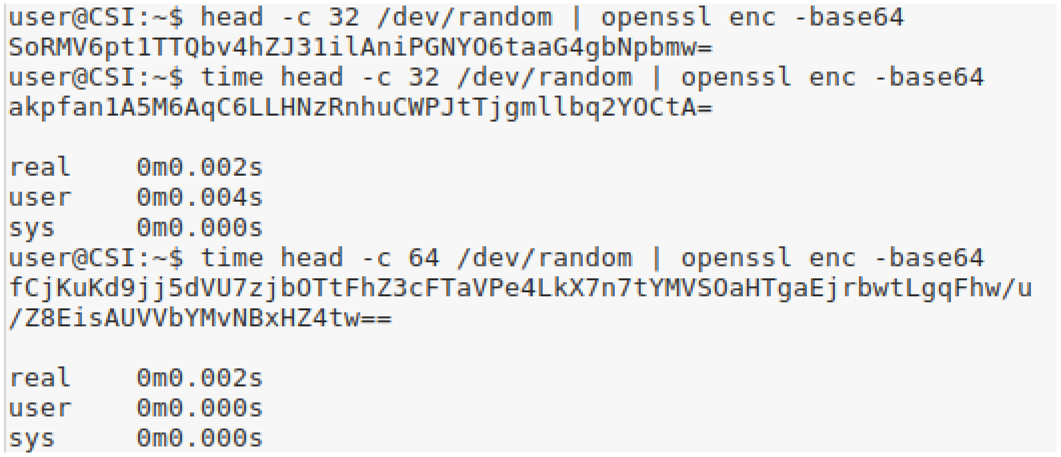
# Conteúdo

# 1 Números aleatórios/pseudoaleatórios

## 1.1 Pergunta 1.1

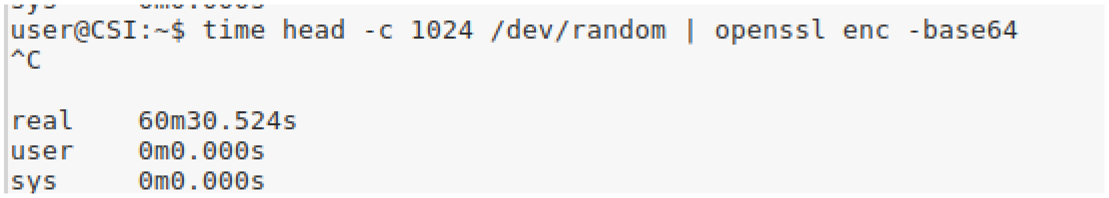
Tendo em conta os comandos presentes, podemos observar que 3 comandos vão ler *bytes* pseudoaleatórios a '/dev/random' e um 1 comando a '/dev/urandom'.

Pela *man page*[1], podemos ver a diferença entre estes dois geradores é que no caso do primeiro irá bloquear caso não exista entropia suficiente ao contrário do segundo. Consequentemente é de esperar que os tempos de execução sejam mais rápidos no uso do '/dev/urandom'. Para verificarmos os tempos de execução, utilizamos os seguinte comandos:



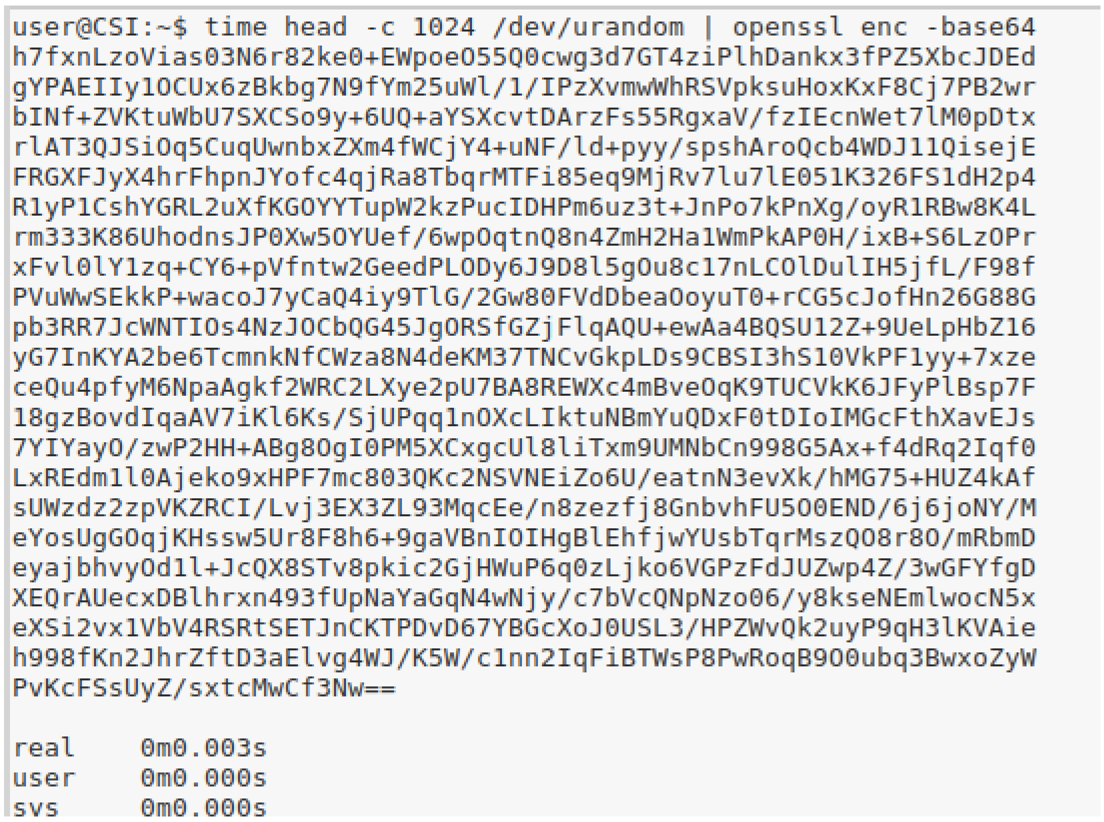
```
user@CSI:~$ head -c 32 /dev/random | openssl enc -base64
SoRMV6pt1TTQbv4hZJ31ilAniPGNYO6taaG4gbNpbmw=
user@CSI:~$ time head -c 32 /dev/random | openssl enc -base64
akpfan1A5M6AqC6LLHNzRnhuCWPJtTjgmllbq2YOCtA=

real    0m0.002s
user    0m0.004s
sys     0m0.000s
user@CSI:~$ time head -c 64 /dev/random | openssl enc -base64
fCjKuKd9jj5dVU7zjbOTtFhZ3cFTaVPe4LkX7n7tYMVSOaHTgaEjrbwtLgqFhw/u
/Z8EisAUVVbYMvNBxHZ4tw==

real    0m0.002s
user    0m0.000s
sys     0m0.000s
```

Figura 1



```
user@CSI:~$ time head -c 1024 /dev/random | openssl enc -base64
^C

real    60m30.524s
user    0m0.000s
sys     0m0.000s
```

Figura 2



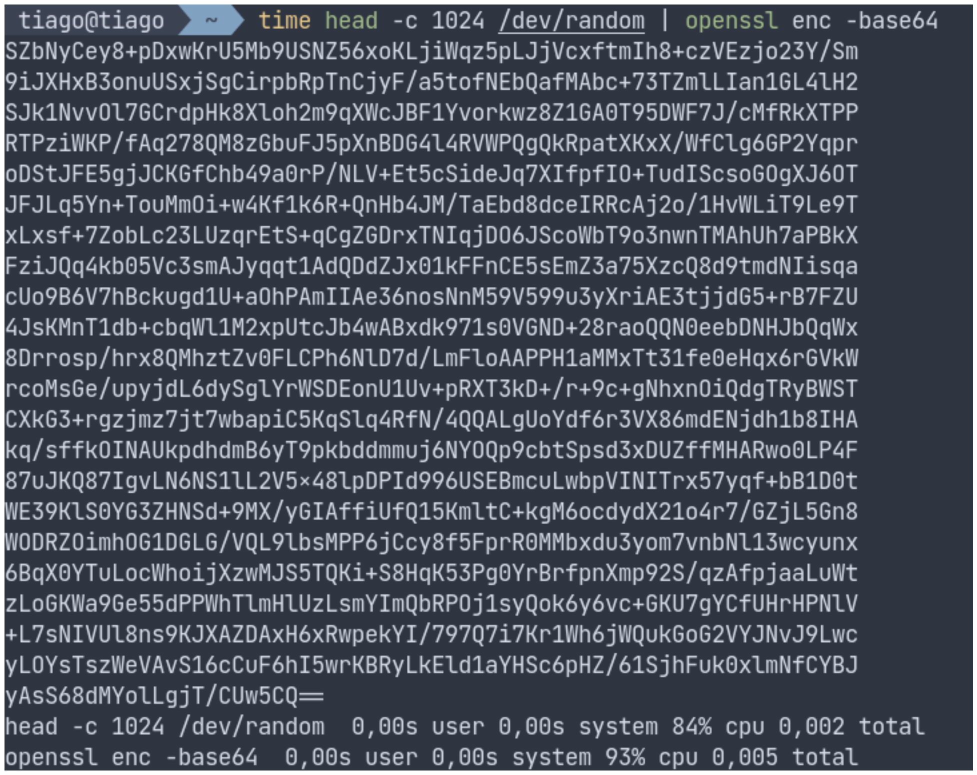
```
user@CSI:~$ time head -c 1024 /dev/urandom | openssl enc -base64
h7fxnLzoVias03N6r82ke0+EWpoeO55Q0cwg3d7GT4ziPlhDankx3fPZ5XbcJDEd
gYPAEIIy1OCUx6zBkbg7N9fYm25uWl/1/IPzXvmwWhRSVpksuHoxKxF8Cj7PB2wr
bINf+ZVKtuWbU7SXCSo9y+6UQ+aYSXcvtDArzFs55RgxaV/fzIEcnWet7lM0pDtx
rlAT3QJSiOq5CuqUwnbxZXm4fWCjY4+uNF/ld+pyy/spshAroQcb4WDJ11QisejE
FRGXFJyX4hrFhpnJYofc4qjRa8TbqrMTFi85eq9MjRv7lu7lE051K326FS1dH2p4
R1yP1CshYGRL2uXfKGOYYTupW2kzPucIDHPm6uz3t+JnPo7kPnXg/oyR1RBw8K4L
rm333K86UhodnsJP0Xw5OYUef/6wpOqtnQ8n4ZmH2Ha1WmPkAP0H/ixB+S6LzOPr
xFvl0lY1zq+CY6+pVfntw2GeedPLODy6J9D8l5gOu8c17nLCOlDulIH5jfL/F98f
PVuWwSEkkP+wacoJ7yCaQ4iy9TlG/2Gw80FVdDbeaOoyuT0+rCG5cJofHn26G88G
pb3RR7JcWNTIOs4NzJOCbQG45JgORSfGZjFlqAQU+ewAa4BQSU12Z+9UeLpHbZ16
yG7InKYA2be6TcmnkNfCWza8N4deKM37TNCvGkpLDs9CBSI3hS10VkPF1yy+7xze
ceQu4pfyM6NpaAgkf2WRC2LXye2pU7BA8REWXc4mBveOqK9TUCVkK6JFyPlBsp7F
18gzBovdIqaAV7iKl6Ks/SjUPqq1nOXcLIktuNBmYuQDxF0tDIoIMGcFthXavEJs
7YIYayO/zwP2HH+ABg8OgI0PM5XCxgcUl8liTxm9UMNbCn998G5Ax+f4dRq2Iqf0
LxREdm1l0Ajeko9xHPF7mc803QKc2NSVNEiZo6U/eatnN3evXk/hMG75+HUZ4kAf
sUWzdz2zpVKZRCI/Lvj3EX3ZL93MqcEe/n8zezfj8GnbvhFU5O0END/6j6joNY/M
eYosUgGOqjKHssw5Ur8F8h6+9gaVBnIOIHgBlEhfjwYUsbTqrMszQO8r8O/mRbmD
eyajbhvyOd1l+JcQX8STv8pkic2GjHWuP6q0zLjko6VGPzFdJUZwp4Z/3wGFYfgD
XEQrAUecxDBlhrxn493fUpNaYaGqN4wNjy/c7bVcQNpNzo06/y8kseNEmlwocN5x
eXSi2vx1VbV4RSRtSETJnCKTPDvD67YBGcXoJ0USL3/HPZWvQk2uyP9qH3lKVAie
h998fKn2JhrZftD3aElvg4WJ/K5W/c1nn2IqFiBTWsP8PwRoqB9O0ubq3BwxoZyW
PvKcFSsUyZ/sxtcMwCf3Nw==

real	0m0.003s
user	0m0.000s
sys	0m0.000s
```

Figura 3

Como esperado o comando 'head -c 1024 /dev/urandom — openssl enc -base64' demorou bastante tempo e mesmo assim não se obteve 1024 *bytes* pseudoaleatórios. Para os casos de 32 e 64 *bytes* foi mais rápido, uma vez que se pede uma menor quantidade de *bytes* e por isso é normal existir entropia suficiente para a quantidade solicitada.

## 1.2 Pergunta 1.2

Com este *daemon* de entropia o tempo de execução melhorou, como podemos observar pelas imagens seguintes, isto deve-se ao facto deste gerar *streams* de número aleatórios a partir de eventos de *hardware*, assim quando o '/dev/random' não possuir entropia suficiente a sua *pool* irá ser preenchida pelo *daemon*, por conseguinte terá menor probabilidade de bloquear.

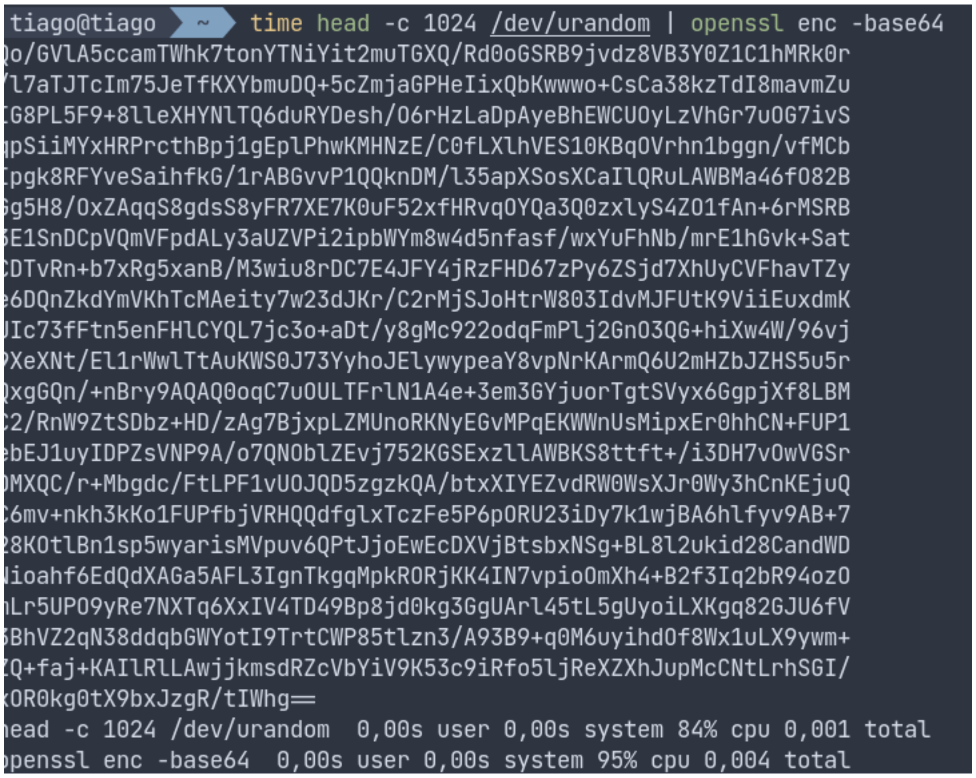


```
 tiago@tiago  ~  time head -c 1024 /dev/random | openssl enc -base64
```

```
SZbNyCey8+pDxwKrU5Mb9USNZ56xoKLjiWqz5pLJjVcxftmIh8+czVEzjo23Y/Sm
9iJXHxB3onuUSxjSgCirpbRpTnCjyF/a5tofNEbQafMAbc+73TZmlLIan1GL4lH2
SJk1NvvOl7GCrdpHk8Xloh2m9qXWcJBF1Yvorkwz8Z1GA0T95DWF7J/cMfRkXTPP
RTPziWKP/fAq278QM8zGbuFJ5pXnBDG4l4RVWPQgQkRpatXKxX/WfClg6GP2Yqpr
oDStJFE5gjJCKGfChb49a0rP/NLV+Et5cSideJq7XIfpfIO+TudIScsoGOgXJ6OT
JFJLq5Yn+TouMmOi+w4Kf1k6R+QnHb4JM/TaEbd8dceIRRcAj2o/1HvWLiT9Le9T
xLxsf+7ZobLc23LUzqrEtS+qCgZGDrxTNIqjDO6JScoWbT9o3nwnTMAhUh7aPBkX
FziJQq4kb05Vc3smAJyqqt1AdQDdZJx01kFFnCE5sEmZ3a75XzcQ8d9tmdNIisqa
cUo9B6V7hBckugd1U+aOhPAmIIAe36nosNnM59V599u3yXriAE3tjjdG5+rB7FZU
4JsKMnT1db+cbqWl1M2xpUtcJb4wABxdk971s0VGND+28raoQQN0eebDNHJbQqWx
8Drrosp/hrx8QMhztZv0FLCPh6NlD7d/LmFloAAPPH1aMMxTt31fe0eHqx6rGVkW
rcoMsGe/upyjdL6dySglYrWSDEonU1Uv+pRXT3kD+/r+9c+gNhxnOiQdgTRyBWST
CXkG3+rgzjmz7jt7wbapiC5KqSlq4RfN/4QQALgUoYdf6r3VX86mdENjdh1b8IHA
kq/sffkOINAUkpdhdmB6yT9pkbddmmuj6NYOQp9cbtSpsd3xDUZffMHARwo0LP4F
87uJKQ87IgvLN6NS1lL2V5×48lpDPId996USEBmcuLwbpVINITrx57yqf+bB1D0t
WE39KlS0YG3ZHNSd+9MX/yGIAffiUfQ15KmltC+kgM6ocdydX21o4r7/GZjL5Gn8
WODRZOimhOG1DGLG/VQL9lbsMPP6jCcy8f5FprR0MMbxdu3yom7vnbNl13wcyunx
6BqX0YTuLocWhoijXzwMJS5TQKi+S8HqK53Pg0YrBrfpnXmp92S/qzAfpjaaLuWt
zLoGKWa9Ge55dPPWhTlmHlUzLsmYImQbRPOj1syQok6y6vc+GKU7gYCfUHrHPNlV
+L7sNIVUl8ns9KJXAZDAxH6xRwpekYI/797Q7i7Kr1Wh6jWQukGoG2VYJNvJ9Lwc
yLOYsTszWeVAvS16cCuF6hI5wrKBRyLkEld1aYHSc6pHZ/61SjhFuk0xlmNfCYBJ
yAsS68dMYolLgjT/CUw5CQ==
```

```
head -c 1024 /dev/random  0,00s user 0,00s system 84% cpu 0,002 total
openssl enc -base64  0,00s user 0,00s system 93% cpu 0,005 total
```

Figura 4



```
tiago@tiago  ~  time head -c 1024 /dev/urandom | openssl enc -base64
Qo/GVlA5ccamTWhk7tonYTNiYit2muTGXQ/Rd0oGSRB9jvdz8VB3Y0Z1C1hMRk0r
Yl7aTJTcIm75JeTfKXYbmuDQ+5cZmjaGPHeIixQbKwwwo+CsCa38kzTdI8mavmZu
IG8PL5F9+8lleXHYNlTQ6duRYDesh/O6rHzLaDpAyeBhEWCUOyLzVhGr7uOG7ivS
qpSiiMYxHRPrcthBpj1gEplPhwKMHNzE/C0fLXlhVES10KBqOVrhn1bggn/vfMCb
Ipgk8RFYveSaihfkG/1rABGvvP1QQknDM/l35apXSosXCaIlQRuLAWBMa46fO82B
Gg5H8/OxZAqqS8gdsS8yFR7XE7K0uF52xfHRvqOYQa3Q0zxlyS4Z01fAn+6rMSRB
3E1SnDCpVQmVFpdALy3aUZVPi2ipbWYm8w4d5nfasf/wxYuFhNb/mrE1hGvk+Sat
CDTvRn+b7xRg5xanB/M3wiu8rDC7E4JFY4jRzFHD67zPy6ZSjd7XhUyCVFhavTZy
e6DQnZkdYmVKhTcMAeity7w23dJKr/C2rMjSJoHtrW803IdvMJFUtK9ViiEuxdmK
UIc73fFtn5enFHlCYQL7jc3o+aDt/y8gMc922odqFmPlj2GnO3QG+hiXw4W/96vj
9XeXNt/El1rWwlTtAuKWS0J73YyhoJElywypeaY8vpNrKArmQ6U2mHZbJZHS5u5r
QxgGQn/+nBry9AQAQ0oqC7uOULTFrlN1A4e+3em3GYjuorTgtSVyx6GgpjXf8LBM
C2/RnW9ZtSDbz+HD/zAg7BjxpLZMUnoRKNyEGvMPqEKWWnUsMipxEr0hhCN+FUP1
ebEJ1uyIDPZsVNP9A/o7QNOblZEvj752KGSExzllAWBKS8ttft+/i3DH7vOwVGSr
0MXQC/r+Mbgdc/FtLPF1vUOJQD5zgzkQA/btxXIYEZvdRW0WsXJr0Wy3hCnKEjuQ
C6mv+nkh3kKo1FUPfbjVRHQQdfglxTczFe5P6pORU23iDy7k1wjBA6hlfyv9AB+7
28KOtlBn1sp5wyarisMVpuv6QPtJjoEwEcDXVjBtsbxNSg+BL8l2ukid28CandWD
Nioahf6EdQdXAGa5AFL3IgnTkgqMpkRORjKK4IN7vpioOmXh4+B2f3Iq2bR94ozO
nLr5UPO9yRe7NXTq6XxIV4TD49Bp8jd0kg3GgUArl45tL5gUyoiLXKgq82GJU6fV
3BhVZ2qN38ddqbGWYotI9TrtCWP85tlzn3/A93B9+q0M6uyihdOf8Wx1uLX9ywm+
ZQ+faj+KAIlRlLAwjjkmsdRZcVbYiV9K53c9iRfo5ljReXZXhJupMcCNtLrhSGI/
kOR0kg0tX9bxJzgR/tIWhg==
head -c 1024 /dev/urandom  0,00s user 0,00s system 84% cpu 0,001 total
openssl enc -base64  0,00s user 0,00s system 95% cpu 0,004 total
```

Figura 5

# 2 Partilha/Divisão de segredo (Secret Sharing/Splitting

## 2.1 Pergunta 2.1

### 2.1.1 A)

Para dividir o segredo, executamos o programa 'createSharedSecret-app.py', que recebe como argumentos number_of_shares que é o número de partes que queremos dividir o segredo, quorum que é o número de parte necessárias para reconstruir o segredo e a chave privada que tivemos que gerar através do comando '**openssl genrsa -aes128 -out mykey.pem 1024'**, que irá servir para assinar cada componente que são objetos JSON Web Signature.

Para reconstruir o segredo tivemos de passar como argumento o certificado que irá servir para verificar verificar as assinaturas de cada objeto JSON Web Signature.

### 2.1.2 B)

O **recoverSecretFromComponents-app.py** apenas precisa do número de quorum estabelecido na construção do SharedSecret (que neste caso são 5), no mínimo, para reconstruir o segredo, enquanto no outro precisamos de todas as componentes em que o segredo foi dividido.

Um dos exemplos em que se pode usar o **recoverSecretFromAllComponents-app.py** em vez do **recoverSecretFromComponents-app.py** é no capítulo da segurança, onde supomos que um atacante possui o quorum ( número de partes necessárias para reconstruir o segredo), mas neste caso precisa de todas as componentes para obter este mesmo segredo. Também poderá ser necessário se alguma operação for crítica e necessitar assim da presença de todas as componentes.

# 3 Algoritmos e tamanhos de chaves

## 3.1 Pergunta 4.1

A entidade de certificação usada foi a **UZI-register Zorgverlener CA G3**, sendo usado o segundo certificado pois era o mais recente.

Segue-se o resultado do comando '**openssl x509 -in cert.crt -text -noout**':

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            61:b6:fd:9d:33:fe:aa:70:f0:2e:27:fb:98:b7:6f:82:82:ba:41:28
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = NL, O = Staat der Nederlanden, CN = Staat der Nederlanden
        Organisatie Persoon CA - G3
        Validity
            Not Before: Apr 18 08:33:53 2019 GMT
            Not After : Nov 12 00:00:00 2028 GMT
        Subject: C = NL, O = CIBG, organizationIdentifier = NTRNL-50000535,
        CN = UZI-register Zorgverlener CA G3
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (4096 bit)
                Modulus:
                    00:d3:93:62:34:e5:c5:62:b8:5b:7e:27:eb:9c:e6:
                    5a:b1:65:8b:d8:b5:dd:c1:35:95:77:7b:a0:b5:10:
                    87:27:44:ed:8f:b3:64:e7:b6:29:f5:cf:32:ae:29:
                    18:97:da:d3:f1:04:2e:9a:6d:e0:59:67:cb:b1:81:
                    4e:77:3f:37:e1:39:cd:c1:d5:f1:16:b0:86:2f:2a:
                    9a:81:d2:9b:e2:d8:dc:9e:9c:27:32:96:19:82:a7:
                    7b:35:7c:08:83:83:df:1e:c3:c4:d0:55:dc:1c:64:
                    c9:e8:17:17:e8:30:81:d0:24:15:e0:0e:ff:57:5d:
                    55:35:ea:d0:18:d6:42:f1:0e:91:f2:6e:43:75:68:
                    ed:bb:06:e7:e6:26:c8:10:63:5e:ba:07:37:6c:4b:
                    19:6b:5d:59:92:48:8d:e9:bb:3a:f0:ba:91:83:09:
                    91:db:73:fc:0b:1f:2c:51:6a:0b:82:a8:29:8c:ac:
```

```
                    49:9d:55:0f:72:31:4a:3b:a5:73:ce:fc:bc:3a:40:
                    b9:06:51:2c:14:d4:c9:02:fe:6e:53:20:80:50:f0:
                    59:9a:7a:31:4b:07:15:46:8d:79:65:96:d2:93:5f:
                    4b:d5:53:7e:60:58:75:25:31:13:04:42:7b:05:26:
                    d0:41:41:ac:a5:31:9d:23:91:90:3d:7c:ab:93:21:
                    75:5b:7b:b4:1d:f7:ce:fc:3f:4a:54:18:a4:5f:10:
                    66:9f:d3:47:8b:97:73:81:28:6d:88:91:cf:dd:76:
                    51:19:13:99:c0:f5:bf:b7:04:4c:53:87:89:24:32:
                    1a:b8:7d:05:4d:eb:6a:01:1d:cc:18:7a:08:64:1a:
                    ac:b7:0b:16:10:3d:5c:e4:b1:90:dd:b1:17:c4:7d:
                    8c:13:dd:c4:81:c3:24:13:2f:f7:8d:57:a8:ae:b0:
                    7e:b2:a7:ef:75:8d:01:f4:87:a3:ba:67:88:f9:f6:
                    e3:69:66:8a:fa:9e:95:30:62:48:28:9c:af:af:23:
                    62:41:e6:4e:94:98:58:fa:17:09:6e:8b:30:73:3e:
                    70:92:b0:4a:ff:3a:bb:3b:85:b8:83:23:4a:47:95:
                    bb:28:a2:34:1e:27:96:f2:96:dc:4a:a0:13:58:31:
                    a2:53:b1:40:4f:1a:82:5c:e9:10:d5:d8:80:64:a7:
                    ca:5e:24:4a:9f:34:92:98:a2:51:a9:03:1f:47:58:
                    2e:76:e1:30:62:fc:be:1a:f4:ef:37:bd:99:4d:47:
                    ee:f8:29:ba:60:4f:fe:3f:b4:89:d5:94:44:0f:41:
                    a5:53:5f:b8:36:75:01:e9:e2:ae:c3:ae:89:81:dc:
                    c5:f1:a9:39:9f:d7:9d:69:dd:02:2e:d8:91:0b:8d:
                    e2:d4:7f
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            Authority Information Access:
                CA Issuers - URI:http://cert.pkioverheid.nl/
                DomOrganisatiePersoonCA-G3.cer

            X509v3 Subject Key Identifier:
                C4:FC:77:D5:08:48:98:12:87:B7:F6:51:13:C5:CB:AE:FB:35:83:4A
            X509v3 Basic Constraints: critical
                CA:TRUE, pathlen:0
            X509v3 Authority Key Identifier:
                keyid:EE:AC:6D:40:EA:D5:04:6A:87:2C:55:7B:F5:3F:2D:DA:EE:DB:AC:E2

            qcStatements:
                0.0...+.......0.......I..
            X509v3 Certificate Policies:
                Policy: 2.16.528.1.1003.1.2.5.1
                Policy: 2.16.528.1.1003.1.2.5.2
                  CPS: https://cps.pkioverheid.nl
                Policy: 2.16.528.1.1003.1.2.5.3

            X509v3 CRL Distribution Points:

                Full Name:
                  URI:http://crl.pkioverheid.nl/
                  DomOrganisatiePersoonLatestCRL-G3.crl

            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Extended Key Usage:
                TLS Web Client Authentication, E-mail Protection,
                1.3.6.1.4.1.311.10.3.12, Microsoft Encrypted File System,
                OCSP Signing
    Signature Algorithm: sha256WithRSAEncryption
         ac:91:93:fa:30:ea:84:30:81:4e:9f:36:d6:80:dc:11:cf:0f:
         36:9b:45:27:f5:f3:49:f4:a1:4d:7e:75:8d:50:b9:c7:cf:96:
         e0:8e:a1:f6:d0:e7:21:83:68:64:e2:28:63:48:4e:9a:72:67:
         0a:f8:09:f5:10:a8:be:9f:a3:0f:12:79:0c:6c:ce:0f:87:97:
         c3:53:94:1e:22:15:3c:a1:65:b2:6d:1a:34:dd:e6:69:78:b7:
         45:a0:16:09:0e:da:8c:5f:d7:dc:f4:00:57:9d:01:1c:05:c7:
         75:e2:58:79:82:4d:3d:c5:56:75:e7:36:85:8d:10:3e:7e:b1:
         e8:86:1a:19:bb:b4:32:b8:55:b6:c9:e9:16:e9:3d:15:a3:ac:
         fe:04:56:5b:ae:5d:a4:c4:6a:af:0d:1e:5e:68:20:13:98:5d:
         b9:e6:3b:22:a6:eb:13:98:fe:44:ff:18:ad:da:14:40:ef:09:
         e4:37:60:0f:cb:ef:20:2c:fa:93:de:b1:e9:9d:0c:8d:af:97:
         fb:6c:f6:92:47:69:fb:37:30:7a:c9:ca:13:13:b1:a8:1c:2b:
         34:ed:cd:e4:26:3b:40:23:34:5d:9d:ba:3d:71:a6:85:4a:ad:
         31:47:c0:d6:a9:d6:10:e5:d2:47:3b:1a:d9:23:8c:c1:f4:b8:
         be:02:71:ca:e5:59:83:18:69:24:f6:a8:7c:b2:e4:7e:24:93:
         4b:0e:3a:7d:56:30:3c:49:63:1a:e0:07:a3:58:18:8d:a3:2c:
         cb:5f:55:d2:57:02:38:28:83:f7:24:6a:e6:4f:d9:53:f5:fc:
         3d:8e:34:ed:9b:f9:bd:3d:3f:ec:e0:d2:99:42:de:cb:35:bd:
         31:13:85:d8:85:48:d9:7a:43:36:e3:f1:ff:de:43:ca:98:07:
```

```
8e:30:29:d7:cd:f8:a7:cf:4e:37:3c:23:34:97:6b:47:bd:d6:
75:1f:d2:7c:a4:0b:7d:78:d1:b5:f2:4c:c8:49:e2:11:ce:da:
72:43:f2:cb:51:1d:79:7b:cf:84:0a:ba:b0:1b:5e:ea:91:9d:
52:b4:49:6c:f4:92:0c:4b:65:b1:85:61:e7:70:90:42:29:f1:
64:f3:0a:1d:83:4a:62:34:29:63:54:df:83:ce:f9:23:3b:32:
d0:a3:99:01:c6:80:ff:6f:2e:45:ae:df:de:5a:d4:26:df:8d:
23:f1:e6:93:58:0c:04:82:a8:5d:2f:2b:2f:3a:65:bd:5c:6d:
d4:a7:f0:25:43:c2:bb:ec:19:cc:00:93:48:dd:9a:32:8d:64:
fc:69:c4:ce:af:0e:8f:cd:76:a8:30:ca:50:d6:19:d6:b2:d6:
3d:cf:77:e7:90:2b:ea:62
```

**Algoritmos utilizados**:

- Algoritmo de Chave-Pública: RSA Tamanho da chave: 4096 bits;
- Algoritmo de assinatura: SHA256 com RSA.

Tendo em conta a validade do certificado que é válido até 2028 e os algoritmos utilizados, de acordo com as recomendações do NIST[2] o sha256 continuará adequado até 2030 e o RSA com tamanho de chave 4096 bits também continuará adequado, uma vez que o NIST recomenda até 2030 para o problema de fatorização inerente ao RSA um tamanho de 3072 bits.

## Referências

[1] https://linux.die.net/man/4/urandom

[2] https://www.keylength.com/en/4/