# Aula 12-fev-2018

## Grupo 8

### 1. Números Aleatórios/Pseudoaleatórios

### Experiência 1.1

Executando o comando “openssl rand -base64 1024”, obtemos:

```
root@CSI:~# openssl rand -base64 1024
9BrOCqlm3Y3oHPokkDgwIqkBIWLwyU1gh7f6HKcQl+TT4necqOyl2sdm3r9viujW
0caefHHePqZyik4wjLrVXIQRImoFZZexLsPZLM1x2i3qMsv0Vm4eWf74PgL9S902
/Rehblb/BbfeszZI5IuylmEgAtAFNO89Zc5jltr3yDzweWEFvH/MfNM9NIrVzstU
r1UXogo0Ogh2IJX5+ZQlMvDvZn/ZUrQCex+IpkNhZs4cBegYo7XFCrAKkawNgAUb
T09Z0zyFeCm0ORWlJlKmLv4xBHjxYKRT+xdLWPFkpMdQT/qG+pOb6OWWk7gXCZfF
9pjqTEV0iu3dNUKoGd9LZKXU4UocN8l9HnKvqSB+onDHVD0gjcjjg9m2hDBrMIMW
IYr3hAYq3QXD4LMRFiFjq5y+cQmh6p2LQKbEhIviWsyRFBhyFRP8bYSzQqojkGCE
hht0nCLdiWG8xElUkGU2aA6BG56Pjc1FrtWv+tqpfpwclVL+srDUnWzn69IzDLh1
RXI+Mb1nRNRDfqahzvtGZJklToseMWu2gOr1B/r2vcSvm5i5vAAL7sD2ip2QqWeu
qU4pgRC/tkwydE44ZH5+bq6IMsl4gtdTuBHEQh1FhdPlOb66RXWPRhn02eUtmuJi
2fHOIg9cGtE0bE5gHTYzJ+e/4mwavkd3dO81OtEjubMr1r4HU1UmAfi6ROCDlDDJ
UwdlbWcQKDiqJy+92JSsjYSMxvz9q2FM1+NqWMeEokataggKJqBQpGD4Zu2cs5oW
IUJHzQGjFb67+VzqLIyW/IhmGYRfJRtTuuuqkDQQp4grlELMQOczmp+YrhScPGYa
6epmrNodi5WOqoCmG9Q1WSR5CyCC0qQch6+PXS8XiB+mPxTOBLQZOD/WXpS+6A5i
X3T8j7kr+uJqUeHLdKnYDJWK/ARTQ0y76OVGJPHvfmkhG+9WbVYnhnjT6NbKxE11
DTuYGokVCgWxv1lom3dNHetfjpF03wGGH66r1rscJCTCU88kONJuzPxfScdfx+XJ
6P9xguePqiO4He6QGYULNTqgHL1CgxAA0I80V1DQMz4EB11ril1OJIGYgC1DMw6P
1PGBy0uEaCCeJzYu6pDSpMze3eMwBdSMpbhQ/QKHcN9EitOSCuKt9SpTsB+9T2ud
p+b8IHIcJimqJXuXH0ClLnrrri+zAtLLGNNNxmM0+pVFmroSaD4wdvObXzkJYH27
g6SlhdtuDCsCB6z1UoCpSfcX4Ws+DqEChcOpRjppsMhtjx6q3n0xf2M9trihXu1i
hyJfOaDoHf0OrOXBvigBk78TfyEN2aiPB35wXBkChIpBWEA9gYSSYrHhnTH/oxZs
Ss3xYl3MnVbWA8HWdLEI6A==
```

### Pergunta P1.1

```
root@CSI:~# head -c 128 /dev/random | openssl enc -base64
YYbiBFXESVVpkfK7r5dH/wxrWaXIlfH+CeYbjB5QR4Y0+OMtcwig96vSuCnoSU5/
QeISljP+TADVX1T9LazdSwtInXTaqTcz0MD9xlwpalTyihbvpH8qhFEE20ueOvwl
+h9CSPV7qKJyEH3A7PcUIyaWv/PMFkf3mQ5tpac06kk=
root@CSI:~# head -c 128 /dev/urandom | openssl enc -base64
ooFRsTZvnlzLu1Cjby4eMDmL4fRbMMis2b81/3AhoGtW1SLZ83Zwd1miJpGgOME7
+8I5AtGn0MBJTggBig+0rc9JLn8zqqSntfuBBrdaOjmtE+7i3ejrao3Buh5q+Pba
O1gwXD/CX85eA+lLWFDsCCaSyYvDioSyzhYzKSbEw4o=
```

Ambos utilizam o mesmo CSPRNG, mas o modo de funcionamento difere de um para o outro. /dev/random bloqueia se não houver entropia suficiente para gerar a aleatoriedade necessária, ao passo que o /dev/urandom irá gerar uma resposta imediata.

Existem situações onde um ou outro convém mais, e ainda situações em que nenhum convém. Quando está a ligar-se o computador, não deve utilizar-se o /dev/urandom já que a *pool* de entropia está mais ou menos deterministicamente preenchida.

Na eventualidade de se querer gerar um *one-time-pad* para geração de segurança da informação teoricamente segura, não deve utilizar-se qualquer uma das funções, já que como referido, ambas utilizam um gerador de números pseudo-aleatórios.

## Pergunta P1.2

```
root@CSI:~# head -c 1024 /dev/urandom | openssl enc -base64
Ik/i0xNimq3u0/TKjrfTWvAf5Hu2Lt/Afo/f3UyWjrBtSK5hmpDRqAVV/RoqBqNq
UfyRsgG4L5fWxVC+fPV01fyGsQD77/BhI5bk+Q4cDzwJCufyw9gqcvr7FNdXQYZQ
mlBsENBON8+zz4WMxOVFUTrv8g4ZQTTv75BTnrYdKcxqEotUaQqnSwBmTGEK2w4s
u0nRYqWWSDgdcNaJN/K8HM+YG62QwzzkjovNNn+evPjqeAVot9gX9CfGr1y5iztZ
dw5LIFfjxzlVH9cxGCDdxvXZUs6MeiS2nDjQLMM+XS4DU6I0N7U7j7lYJUMqfNVh
qY0buF+ueGRpWM9B8kr0q6jS6R78tBP0fiMTav0mEJJohjqTF2Iu/ooPTvV95Mbo
yGYaFAuXFxHLE0z9rzrJZhcOatqB2THzgKXgEUgzcfT8WweWHqfnstinrYgw8BSs
9Bo57CHPuEmOT0AK2dkiluUPWIZCk5Ia/utB7ljJoROg6gxZACfi8QouScLkmvNU
gFQzFrqNHIGyw/k20YvoVuK/+Dz1WGWTMfoA20TIefZIaupIhmo3p/k7l4Z92cxh
jBYZN5gzp1eJmE7ZSSYN/xFBj8vRsxIeP6CDjvBBN+AGmgah6Xw9J3fXGgLFxO26
CEOUoQcdsUaxQG+GJVfD6fCehlpXLbXUHOujT0PBBs4blQ9Stf50tRuvix/AvYhJ
x4cHsDp+VTyUf3Xw0w83R00sZGumImJnwTNUMNCk5hP/MZbi7CSLgPnA437inEUx
p2aybiaRxA5WE/7quKPDrXtxdyEzzNR4zTKy7JOOyo/gOl/+HEA9Oi1dsm80/E9X
zPaALfMcChXhWi5BlBUUAhBNx2iy28J3Jg6HyZHB5nd/MYWsqL7nxGuMcIen/9zE
5TXb5IgHFKuG+5VVterTQKXcZqNsJhzYsG7oGMdkRIOVH+O6hCdVezRlyKDNBmLN
djvKQQ8mt7roIxFDgWo8AYkuhq3giVPMmB67QMYqTZ6/xHet6IJ4nBk/aZ0otfpN
G+XhEASNz3DljSBk9KcBAEBryBjz3vIu6Lf143DKCW6W2fLix1M+E/uCBm7VzRO6
TnVXRtpGiDi/d0AcjTZ2CaUAvvSHmzPYQTHw/K8nly3ZiedYNQWnilgMnWI9fli9
l6+fa7PzpnLllpnFGj1ex6iQPp9nKC/tKGbA47Fij0t+tYY8MW70mj0IT+7k24xT
LkIV95/KmtymDirkv8wT/SMmxqvt9yx7r5M7YADvToGr6vkEyUgFWtHSl7wiWwfd
SXusSq4H3R5DzBwYoUDt7bQbOm/EnHmzmglWuyf/Q7GyL311/29tBnTaBH8V1+yA
6tFR+IcglHBjKtU5+usp4A==
```

```
root@CSI:~# head -c 1024 /dev/random | openssl enc -base64
joDmPcfmuxmRZtQyOGRFny+l9ED2tIh0dPKxuFxsc3ed9wIawHUMY9vb1MpEHQz2
4T055sW6xqsQCsYBnLpjyD9fpukSed81UHdFjU2gRFOAiHHUxJpJultaxzIwg2hX
NUf+vzDw2ao4JXIPPTDY+xn+TPXPyl5nbrOFXAQY5L5mS8oMRYJlIT+4tT+X8E7n
mqchYnw7Mfry29kM70vPWA+GPFoYcQFkRksd212YZMOK0V34aeZr3+BYHiyj+TPu
Efhs4VT10bqa7XU38LkUokaoFhQyOYi1VUQfJ4NqADSFGLUfErrBneNUWwm4ZOil
6EMi2pJZnS5T2s3//CHXdU/tiFFbulv1Lfi8T0oJH+nypFDPGzC/921eWEYMExkW
57mP1SqsmKErVbYTVgimvNcKJIar6XVRFYxeoFD+bIHIWCACSYYHFdid0R9WMo/t
ejTrWYY5iSQ5ReRGNFGTnYfRJPVp/t2qd2OPG7kXkbaerp4cGUyRXLGjfGWstG3D
ajJ+gI6dSFudFI/oEVcZ12zUYsA48lXzYmcgqDLmgbOpKtmzDgoKAujH67+1RvYL
7z90EI6/0EZQfEUJsn1KnSRyXhLD3EOTyGcRLyWjY9Re6XrN3E5VEbvLKCchRKdA
UyETjZIUtVwQOsuvT0N7XFEIOiWARpS8ksETPoxtl0BqHqqt9Wq2M0OLdtOE1B9E
DXU4hBpqoEVXTIwrYF6WTkqdfQJWVxFl4qsNbnR5RSl+OcnckOsYbdd1KNScZL9d
Ai+VM3IBAY69eIErVU4kykyofI0xma0CULWcHrwGzPCYYSuADqJPvHjFELSKLd47
umjLvYiGopJoRKsgzFJtz+if5T6Ny6Y0ij41P2XLXyVfFGYm7xproaEQXnfR+6U/
jn5wisD9ZX8y/z0Sn/KDfaBEkzGwxpbwIrOPlo/fM8rpWdy2uvUS35pGCk6hZnms
1FWTbzYTa45RG9fvufTqJ6uynrnt+UBqAHXd2eoI0ez9xPGZLEYyP5ErLaRI4rvE
XGw9biAFF6b0P791kItdc/jI4elzSzAnPK4ANmaOHVQb7eNSno+lDelPLL8qbkxz
jQXrO+kqHseMxgnDUFeXgqI/kx3wrt7pE0VrovRprWvnjzoCVZPIbJeMXQop3XI6
PqfCDWti0PNikMCiVNq80YX+CNcRzITcvuo42kfCbeI7QVIK2yW9l3lc1KMJQZmw
OZ4pfbyeeZ/P7eh+iTBHEi5RgPjiP/7QQZpuROI93dykpPq8THEhGbDKij5isfRC
PlyfccgPkjAkguo8jjxlW1aurouuYV2ByR/PC84y/PH04j9JbmnAznk+BS7lGA96
KfGWBfTdvRggC/gsKSYCTA==
```

Depois de instalar o haveged, ao voltar a executar os comandos anteriores (agora com 1024 bytes para o /dev/random) foi possível observar que ambos retornam um valor aproximadamente dentro do mesmo tempo. Este é o resultado esperado, uma vez que o haveged serviu inicialmente para colmatar as falhas da fraca geração de entropia por parte do /dev/random.

## Experiência 1.2

Compilando e executando o ficheiro, obtemos o seguinte *output*:

```
root@CSI:~/Desktop/1718-EngSeg/TPraticas/Aula2/PseudoAleatorio# java RandomBytes
 1024
```

### Pergunta P1.3

```
user@CSI:~/Aulas/Aula2/PseudoAleatorio$ python generateSecret-app.py
Usage: python generateSecret-app.py length
user@CSI:~/Aulas/Aula2/PseudoAleatorio$ python generateSecret-app.py 1024
wpKTnOEJuObGJd51siWwfhEKATGTAFcOxvAMOH9a4WFfMOIp7FHcWGYi6b8YS5JznVXIjkXeTwunFjCl
zHXC8QYXWqPGD6XRt3qDCXYPG9JHrABpM8ccKg0jJXJJiFBaR9O951RHraxGgS7pzORtnuAK5gYiRAY0
mx1G4DL4ZB0KqlLiuzDopqUeCLmjDSa1zcL9mDixAjlEV4orgVZPYZXMRQFjyEjoMgwM3NFK8to20fNU
OVM2Asn60o05IqpWqJ8DIRJhgoFsnfiwi5gAxshCin9KSHKYoq52XJQ79IVgxA6NByvil08fM4QbrU6R
MKy2ck3GQQKSEOc55EGxUU05rgBRTMveSY8ErJGEEaHt4cbTV5tO0CYoCicZobKPP6mEUQ51L1GnZtly
9VLp8ySprQJf2gLa0BuJXkgKOxTV8phzpd8EHV5aoipFFcoe7TkYXIF4aqosd4E4A6OtMHt1OK2cHXpT
hyEDE6mIGDzML7oUc9Mr2oNs0PizHSxxed4tf12z1WzRaGLMwfhDNiquKQPR9kAF68Z6EyY3xuwUuYjA
KwiY6k8xYf3Hxw85kqom8J3S7QWlPl366XEFp1mEH1biwdYYLgSRHOiDOnoj4LfX8jEOyFZpPMhotPN7
jT5azRlwrftT70H9bHDmbPCLLZ44qram8h3DKzQSGOMd8nHjiz5KuSH3RiHlT9Nd9XhPAS181jjC7Y8g
DP2WyHagutoKcBArgI6obVjE4PpNuUGFuPF52MblBgxIT1yeaBV3jcWOCQ1G3UNaxisosNagpwV9xCIj
joxlBJWfiD8V1UdpC9sLvNfXX6Rt7B4MdiexTqsa1zh32vBjkiBinqfHnvHPIa26hRoq4hKdlAdLRnx3
dmJiqv1pgxy4W8gSwI4SoOXZ6SJzquaHtr4OpYoGbDsM0qMT3Ms10AeCQkOAy03FLWD4Iz7VDmnt7tFD
YKUkcQR3WyIKKB3VT4F6ctnwQ8nWn9dIALzyD55gNBx5lN6IxkbXBAsHv66702hs
```

```
def main(length):
    sys.stdout.write("%s\n" % shamirsecret.generateSecret(length))
```

```
    while (l < secretLength):
        s = utils.generateRandomData(secretLength - l)
        for c in s:
            if (c in (string.ascii_letters + string.digits)
```

Como podemos verificar, a variável ‘c’ apenas toma valores de letras ASCII ou dígitos.

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

### Experiência 2.1

```
user@CSI:~/Aulas/Aula2/SecretSharing$ php genSharedSecret.php secret 3
Codigo 0: 01101000110001001110001111001000100101101001010 1
Codigo 1: 01100101010010011111011110101100101111101001110 0
Codigo 3: 01111110111010000111011100010110010011010111110 1
user@CSI:~/Aulas/Aula2/SecretSharing$ php reconstroiSecret.php 01101000110001001
110001111001000100101101001010 1 011001010100100111110111101011001011111010011100
 0111111011101000011101110001011001001101011111101
Segredo: secret
```

```
user@CSI:~/Aulas/Aula2/SecretSharing$ php reconstroiSecret.php 01101000110001001
110001111001000100101101001010 1 011001010100100111110111101011001011111010011100
(edo:
```

```
user@CSI:~/Aulas/Aula2/SecretSharing$ php reconstroiSecret.php 01101000110001001
110001111001000100101101001010 1 011001010100100111110111101011001011111010011100
 0111111011101000011101110001011001001101011111101 0111111011101000011101110001011
10010011010111110 1
(edo:
```

Relativamente ao *genSharedSecret.php*, como podemos ver no source code, inicialmente transforma-se a passphrase num array de bits, de seguida são gerados ‘argv[2]’ arrays de random

bits com length definida pelo tamanho da chave em bits. Finalmente faz-se o xor da passphrase com os arrays criados acumuladamente, e imprime-se os respetivos.

Como se pode ver na imagem, o segredo foi dividido em 3 componentes. Quando tentamos reconstruí-lo com menos ou mais do que 3 componentes, podemos ver que não é possível obter o segredo inicial. Isto é apenas possível com o número de componentes indicado inicialmente.

## Experiência 2.2

```
user@CSI:~/Aulas/Aula2/ShamirSharing$ echo secret | perl shares.pl 2 3
2:1:d14f85f0bb2d:
2:2:2e39a76d10e7:
2:3:8c23c9eb66a0:
```

```
user@CSI:~/Aulas/Aula2/ShamirSharing$ perl reconstruct.pl
2:1:d14f85f0bb2d:
2:2:2e39a76d10e7:
secret
```

```
user@CSI:~/Aulas/Aula2/ShamirSharing$ perl reconstruct.pl
2:1:d14f85f0bb2d:
2:2:2e39a76d10e7:
2:3:8c23c9eb66a0:
Ignoring share 3...
secret
user@CSI:~/Aulas/Aula2/ShamirSharing$ perl reconstruct.pl
2:1:d14f85f0bb2d:
too few shares at reconstruct.pl line 77, <STDIN> line 1.
```

Como é possível observar, o software limita a tentativa de recuperação com um número de chaves maior que o quorum especificado na criação e dá um erro de poucos inputs caso a tentativa tenha menor número do que o especificado no quorum.

## Pergunta P2.1

**A)** Para dividir o segredo em 7 partes com quorum de 3, foi executando o script com os parâmetros 7 (number_of_shares), 3 (quorum), 1 (uid) e private-key.pem (private_key).

```
user@CSI:~/Aulas/Aula2/ShamirSharing$ python createSharedSecret-app.py 7 3 1 private-key.pem
Private key passphrase: secret
Secret: Agora temos um segredo muito confidencial
Component: 1
eyJhbGciOiAiUlMyNTYifQ.eyJvYmplY3QiOiBbIjEtM2ViOGU5Yzg1ODZjODY4N2QxMTRlODQxZmUzODkxMmE5NjcyZjVkYzg0NW
I3MWIzYjQzMWU1ZGFjYmVhNzc1N2NmOWMxOWRkYzM1ZTQ1M2MzYzM4OThhNzU5ZTg4YmNhIiwgIjEiLCAzLCA3LCAiNDllN2U5OGU
yNGE5MTdmZDFiODEwNzM0NDJmMTEwN2UzNmMzMzcyZDVlY2VhNzk4MmFiZTNlM2QyMjcwNDA4NyJdfQ.rxvO53ErG4OWKu3mgLwL4
nqiJVw690vbqCzlwNqXZKz5kDO0jOzHNRbWHN_qv4MAQ5KP_5aR1b_mVZXkz4rBdf1_sIGDooArHm1JvaZjwMsSS479UFSIGVqRU_
WiG39qYa2dM40JFfWtclfBjkfmXgBYZC_k-JnukFdfSNzdC_M
```

```
Component: 7
eyJhbGciOiAiUlMyNTYifQ.eyJvYmplY3QiOiBbIjctY2UwM2YwODQyMjJhMDUxNTI3MTJkZDEzNWE5MzhmN2RkMDk5MjAzMjI5YT
UzMDM2NmRkNGI2MjU2ZmVjODZiMjdiMjdhN2JkYWI5YzgwNWY4YjU1ZmIzNDVmM2QxZWYyIiwgIjEiLCAzLCA3LCAiYjNjNWU3MDV
kMmE2MzE1NjQxNmVhMWRhZGFkOWRlZGZhZmQ3ZDI1NGVjYWI0OTNjYzY5NjcwYTgyMGM3OWJkZiJdfQ.dDyCwTm23iNybANLqdWcy
YendyIVJTDlaP1XYinmed2ipqvfORl9f0NpKLvm_-Aw3zx0a4EU68_-ADWsbLMVUiVlsCYPZpOQwZhPPnGJGOGl3xo8CjOIRzDVyS
cGgrID6jBIKqNLR-4PE-OHHJ1-DMWSrYWqfXoTRxDFllgZiP0
```

**B)** Para recuperar o segredo através do recoverSecretFromComponents basta atingir o quorum, como se vê na figura abaixo.

```
root@CSI:~/Documents/ES# python recoverSecretFromComponents-app.py 3 1 mykey.crt
Component 1: eyJhbGciOiAiUlMyNTYifQ.eyJvYmplY3QiOiBbIjEtYTkyNTVhN2M3ZDg3MThiOTdiYWY4NzAzZDgz
YmM1MWQ1YjNjMTgyMjQwYWVhOWU3OWY3MzBlZTcwNTEwYWFlMTkzNmI1NjEzZTAyNDk2YzU4NGUxNDNlYTE5NTFiZTc2
IiwgIjEiLCAzLCA3LCAiMGFjMzJlMjI4YWIwYjNiZDNiNjkyNWE3NzlhNDAxMDk5ZGQ5N2I2ZDMyY2RkMzExNWM0NTVm
YjhlODNjNTg2YyJdfQ.ltInx0KWDDl9tAS5ktyyGOEBWux17hNmfWt55vdas3FkETFkFx3v4x_3CuOZIJweW3NG0YzzJ
QSGiLoJC1Djjs2iVQCxA5T5xDwjmkmyody8PBcJI_KRgkw8mJ_0_k-dpcVaAQjnQ7kaOVKOtOWtpAZYJypOVwx6mt_34
s3O3JE
Component 2: eyJhbGciOiAiUlMyNTYifQ.eyJvYmplY3QiOiBbIjItNGMzM2YyMjFhMjRkMjY3MWI5ZTg0NDA5NmFj
MjRiYTg5YWM4NWYyNWQzNGYwNzlhMGFkNmEyZDE4ZWY4OGE4MGYxNmViZDJlM2ZiOGRkNTc0ZmJmNWQzMWYyYmI2ODVl
IiwgIjEiLCAzLCA3LCAiMjU3YTcxMmZiZWM4ZTU5YzJlMzQ4YjlmZmNmOTEyYmRhZThmNDIzNjU5N2VhNGVhNzRhODUx
NGM0MjZkOWVhMSJdfQ.e-SqnouBQaLI05G98hsrN2ETioaDehtyOkIGGk1iK7PRjS8z4vfIsDnC-qyyEPbKX32Qkohjw
K6DX1v3F9jiVKcfwbrv0yLEq-G3E-tBM8ZKKuz20LG8n8hONWj_YMy092fcpshATm0bJamlYlZDxefAaPuWlNMeADwVf
DNYzCE
Component 3: eyJhbGciOiAiUlMyNTYifQ.eyJvYmplY3QiOiBbIjMtZTkyYmM2ZWY2ZTUyMjk2YTIyMTlhOTcxZDgw
N2Y5MGYyZTE3ZjU4MDI1MDE4YzdjYTk5ZDIxMjQwY2Q4MGM1MzgzN2VhNDZmODIyYzQyMWJjOWZlYjE0NWVmYTY2MGYy
IiwgIjEiLCAzLCA3LCAiYTExZmQzZTVjZDhhNzJmYTFlOGU3ODVhNzZiZmVhOWEwNTlhNjEwMjFjMzgyM2MyZTllMjRk
ZTUxZjZiNTkxMSJdfQ.jtluMeA45mzaQovjjmWT5IEO9m-2TIdYUo8K1u82qTjTGQYpw74JVHU0LInlpfa6u0pX3yvhS
iY20dj9RNFXebTrjTAJ7PYWEM8EQ3fvb5AQ-9MJeR-AUwjGgB5_Y36Ld7aB9fgtdSZZIL4YP-SAQSXpKkfQT3l9LJqz9
S7uTYg
Recovered secret: Agora temos um segredo muito confidencial
```

Quando executamos este ficheiro com um número de componentes superior ao quorum, ainda conseguimos recuperar o segredo:

```
root@CSI:~/Documents/ES# python recoverSecretFromComponents-app.py 4 1 mykey.crt
Component 1: eyJhbGciOiAiUlMyNTYifQ.eyJvYmplY3QiOiBbIjEtYTkyNTVhN2M3ZDg3MThiOTdiYWY4NzAz
ZDgzYmM1MWQ1YjNjMTgyMjQwYWVhOWU3OWY3MzBlZTcwNTEwYWFlMTkzNmI1NjEzZTAyNDk2YzU4NGUxNDNlYTE5
NTFiZTc2IiwgIjEiLCAzLCA3LCAiMGFjMzJlMjI4YWIwYjNiZDNiNjkyNWE3NzlhNDAxMDk5ZGQ5N2I2ZDMyY2Rk
MzExNWM0NTVmYjhlODNjNTg2YyJdfQ.ltInx0KWDDl9tAS5ktyyGOEBWux17hNmfWt55vdas3FkETFkFx3v4x_3C
uOZIJweW3NG0YzzJQSGiLoJC1Djjs2iVQCxA5T5xDwjmkmyody8PBcJI_KRgkw8mJ_0_k-dpcVaAQjnQ7kaOVKOt
OWtpAZYJypOVwx6mt_34s3O3JE
Component 2: eyJhbGciOiAiUlMyNTYifQ.eyJvYmplY3QiOiBbIjItNGMzM2YyMjFhMjRkMjY3MWI5ZTg0NDA5
NmFjMjRiYTg5YWM4NWYyNWQzNGYwNzlhMGFkNmEyZDE4ZWY4OGE4MGYxNmViZDJlM2ZiOGRkNTc0ZmJmNWQzMWYy
YmI2ODVlIiwgIjEiLCAzLCA3LCAiMjU3YTcxMmZiZWM4ZTU5YzJlMzQ4YjlmZmNmOTEyYmRhZThmNDIzNjU5N2Vh
NGVhNzRhODUxNGM0MjZkOWVhMSJdfQ.e-SqnouBQaLI05G98hsrN2ETioaDehtyOkIGGk1iK7PRjS8z4vfIsDnC-
qyyEPbKX32QkohjwK6DX1v3F9jiVKcfwbrv0yLEq-G3E-tBM8ZKKuz20LG8n8hONWj_YMy092fcpshATm0bJamlY
lZDxefAaPuWlNMeADwVfDNYzCE
Component 3: eyJhbGciOiAiUlMyNTYifQ.eyJvYmplY3QiOiBbIjMtZTkyYmM2ZWY2ZTUyMjk2YTIyMTlhOTcx
ZDgwN2Y5MGYyZTE3ZjU4MDI1MDE4YzdjYTk5ZDIxMjQwY2Q4MGM1MzgzN2VhNDZmODIyYzQyMWJjOWZlYjE0NWVm
YTY2MGYyIiwgIjEiLCAzLCA3LCAiYTExZmQzZTVjZDhhNzJmYTFlOGU3ODVhNzZiZmVhOWEwNTlhNjEwMjFjMzgy
M2MyZTllMjRkZTUxZjZiNTkxMSJdfQ.jtluMeA45mzaQovjjmWT5IEO9m-2TIdYUo8K1u82qTjTGQYpw74JVHU0L
Inlpfa6u0pX3yvhSiY20dj9RNFXebTrjTAJ7PYWEM8EQ3fvb5AQ-9MJeR-AUwjGgB5_Y36Ld7aB9fgtdSZZIL4YP
-SAQSXpKkfQT3l9LJqz9S7uTYg
Component 4: eyJhbGciOiAiUlMyNTYifQ.eyJvYmplY3QiOiBbIjQtODAwY2Q4ZTVlMTk2MjFhMmI0NDNiNzNk
MjAwY2NkNTExNTJhZGIzMTM1YzYzODhmN2JjNjg5ZGU3ZWFmMzA1OTQ5OWIwYmQ3YTc3ZWM1MTJmMzlmNDAyNjEw
MTJhNjY0IiwgIjEiLCAzLCA3LCAiNDYwMjg2NjA4NWVkNDZlYThmMWVhZDI3YjQ0ODdlZWM3YWYxMTI0NGVmMmQz
YjFkNzllZWQ0MDFlYTFmOWJjZSJdfQ.r7FyqiGyHCDixAw99HEIR1_Flu5JvYmW2ApkJ4aV7cu7B7tjymjkrlGt_
o3Zyr0ZA-8V7pHgdkEhQ1zskttHWROkTw9HJs5tTRXUnkvs6Horv9Xsw7K48gSfzbRW8SPFOnvENu-VkLz3eSYjK
2s6jmEp_k9v8RenX5x9151dU2k
Recovered secret: Agora temos um segredo muito confidencial
```

Já no caso do recoverSecretFromAllComponents, é necessário passar todas as componentes do segredo, caso contrário é lançado um erro:

```
user@CSI:~$ python recoverSecretFromAllComponents-app.py  7 1 cert.crt
Recovered secret: Agora temos um segredo muito confidencial
```


Como se pode observar no source code do shamirsecret.py, a chamada do recoverSecretFromAllComponents é equivalente à do recoverSecretFromComponents quando o número de componentes é igual ao quorum, caso contrário é lançado um erro. Com isto, esta função é utilizada quando queremos que o segredo só seja revelado na presença de todas as componentes.

## 3. Authenticated Encryption

### Pergunta P3.1

Assumindo que o servidor detém uma estrutura de dados para verificar a validade da subscrição anual do cliente, e ainda que ocorreu o estabelecimento de uma comunicação segura, definimos os seguintes procedimentos para cifrar e decifrar os ficheiros:

**Cifrar**

**server**: if(valid_client(username) && valid_subs(client));
**server**: genK = pbkdf2 ( username || hash(password) );
**client:** genK = pbkdf2 ( username || hash(password) );
**client:** send(segredo_plaintext , etiqueta); //etiqueta pode ser null
**server:** segredo = cifrar(segredo_plaintext);
**server:** insert_into_hashtable(hash(segredo), getDate());
**server:** hmac := hmac(k, segredo+etiqueta);
**server:** return( cyphertext || etiqueta || hmac );

**Decifrar**

**server**: if(valid_client(username || hash(password)) && valid_subs(client));
**client**: clienthmac **=** hmac(k, cyphertext);
**client**: send(cyphertext + hmac);
**server**: if (hmac(kdf (username || hash(password) ) != hmac) return 0;
**server**: plntxt = decifra( cyphertext , getKey(getDay(hash(cyphertext)), keyRepo));
**server**: return( plntxt, hmac(k, plntxt)) ;

Com este algoritmo, asseguramos a confidencialidade através da cifração segura do conteúdo, e a integridade e autenticidade através do HMAC.

## 4. Algoritmos e tamanhos de chaves

### Pergunta P4.1

A informação relativa aos certificados das seguintes Entidades de Certificação estão resumidas na seguinte tabela:

| | Algoritmo | Tamanho de chave (bit) | Validade |
|---|---|---|---|
| Cartão de Cidadão | RSA | 4096 | 15 de set. 2029 |
| ECCE | RSA | 2048 | 23 de jun. 2018 |
| Justiça | RSA | 4096 | 13 de set. 2019 |