



UNIVERSIDADE DO MINHO

DEPARTAMENTO DE INFORMÁTICA

MESTRADO EM ENGENHARIA INFORMÁTICA

CRİPTOGRAFIA E SEGURANÇA DE INFORMAÇÃO

Engenharia de Segurança

Ficha Prática 6

Grupo Nº 3

Ariana Lousada (PG47034) Luís Carneiro (PG46541)
Rui Cardoso (PG42849)

26 de abril de 2022


Capítulo 1

Parte IX: Criptografia aplicada

1.1 Pergunta P.IX.1.1

1. Analise e execute esse programa de geração de segredo aleatório e indique o motivo do output apenas conter letras e dígitos (não contendo por exemplo caracteres de pontuação ou outros).

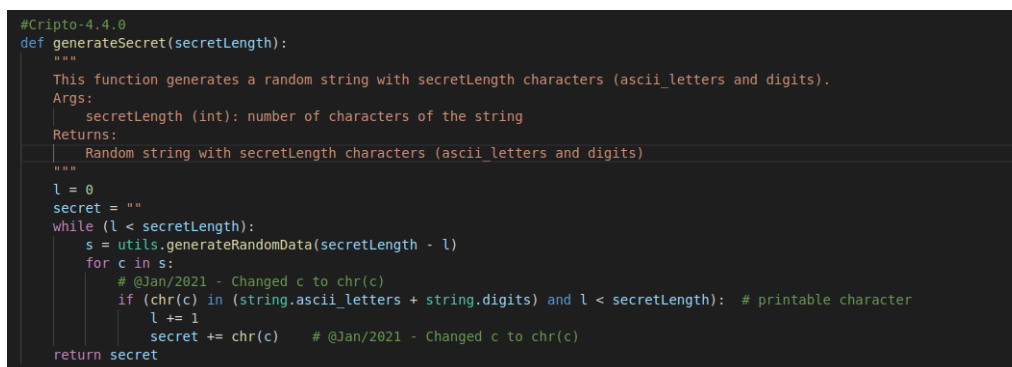
Começando pela execução do código com o argumento '1234' obtém-se o seguinte segredo aleatório:



```
arrow@arrowVM:~/Cripto-py-develop$ python3 generateSecret-app.py 1234
0LTQw6CTOPKa8Z3Q0SRAIW9fkjagNioX60ThvfNosuPd1u0F1rOC9oGg5498789s3YQtF97qmIG9KbPG
aqHVDU0ql4sCo4UmgdfMut68suDjZ0GzKkd0S603rXagL5feuFLWBB4LSOANzuUY18kwqD4JNietITUZ
dF0BSdd158wCpKpdstmnvmsQcsYZFfoKdpYHuhu1kmcCNdk3E0M0uWZFjSZzmCKqZgGw4ozyZmn3KciS
ZU0eKqpvdRrmIfsEAhqVSeTRvLLbh3BbyGdCcMpa1DLSay2knnFnKB9rdcRweB6N0A1ydEaFofST07KS
OtTW269tvShciWH3WgxJxUSH01FNW7nflMH2Rk52cKCQ9MTDzbmwSq05gl2Rj1Ztq4hp49IcULeDQXFg
rOKy29Ya8zG0dLX1ZwwYDbWepbC30sPIqqdrnnMFCEmlycvNPDxzL9rxhcqush5zDZDIAa5aTTTLbha
K4qlJUHqEL7XBLMT3RVZmWEtouVbTaL0uIhU8dY4073rtbsZ5xeXx4cEEEn8TOfBAqBF96GQH5jgARuG
3nx12VevXXOhzpzN7eYdXb638EhpxygykzwYHMYaJcHQVQDLy8F8R1drvvvzfVqwqkHEtxjneLaM0jQL
sNGVsWjhlD0B0Xjarkat9spqQW8udXIcsiQbTss6E79imgFhMdkTQ02Nl8pWzX9biLfQXae6pOZWAdP
zXXq6Axc1Rsta9GnJQ0ohlRs8J0J3htjaPq8m29t7VfDyKbrgT67HyRLYt804s3aNWOMLUYpxX2ptxCf
49wsyoY9XKQ7mA766Teeqh0ErBiYfYZTznZ3aRD8yAeVftklqLbaGgtP9fA38z1RGwHfe3rrrgWDjtgNN
tEttGP0BmPrXqeZ8fmb4F4WAMSbKxjLpVzrhLAxrcAas2ZXmT1DkgGRGM15XQJnJV0jM3w9I7z20Q6b3
V9ZtFOIPmxggPZIwrewt4kS3pKjBgGWH8LSHusBxDpvUNPjI39BZ3bmFZpUwGvY2RLft7GnAJ482pR2q
kPaBPaY35ZLORBERAusjzJQBFR7WH9mGh5WRSi3Bkp9et7xUvN71lpxc6c754ReJxpfiMAEdW66VcI
9QYqIs9Ehe2z8zLsdoIeUStZQ0iaffZ0L5s8CqURk3ZkIbqrhBUH00HImyMwppkqr8f2vNvlo40m9gcJ
vL994SZswg3zyiP2bxZ9FjiYNNvGMD5ZLR
arrow@arrowVM:~/Cripto-py-develop$
```

Figura 1.1: Execução do código do ficheiro `generateSecret-app.py` com o argumento '1234'

Consultando o ficheiro `shamirsecret.py` do módulo `Cripto` utilizado, tendo em especial detalhe a função `generateSecret`, podemos ver que esta função só cria segredos com dígitos e letras:



```
#Cripto-4.4.0
def generateSecret(secretLength):
    """
    This function generates a random string with secretLength characters (ascii_letters and digits).
    Args:
        secretLength (int): number of characters of the string
    Returns:
        Random string with secretLength characters (ascii_letters and digits)
    """
    l = 0
    secret = ""
    while (l < secretLength):
        s = utils.generateRandomData(secretLength - l)
        for c in s:
            # @Jan/2021 - Changed c to chr(c)
            if (chr(c) in (string.ascii_letters + string.digits) and l < secretLength): # printable character
                l += 1
                secret += chr(c) # @Jan/2021 - Changed c to chr(c)
    return secret
```

Figura 1.2: Função `generateSecret` utilizada para gerar segredo de Shamir

Isto acaba por limitar o *output* do ficheiro `generateSecret-app.py` a apenas dígitos e letras.

2. O que teria de fazer para não limitar o output a vogais e dígitos? Altere o código.

De modo a permitir a geração de segredos com mais caracteres, é necessário comentar a condição na função `generateSecret` utilizada:

```
#Cripto-4.4.0
def generateSecret(secretLength):
    """
    This function generates a random string with secretLength characters (ascii_letters and digits).
    Args:
        secretLength (int): number of characters of the string
    Returns:
        Random string with secretLength characters (ascii_letters and digits)
    """
    l = 0
    secret = ""
    while (l < secretLength):
        s = utils.generateRandomData(secretLength - l)
        for c in s:
            # @Jan/2021 - Changed c to chr(c)
            #if (chr(c) in (string.ascii_letters + string.digits) and l < secretLength): # printable character
            l += 1
            secret += chr(c) # @Jan/2021 - Changed c to chr(c)
    return secret
```

Figura 1.3: Função `generateSecret` utilizada para gerar segredo de Shamir

Assim, se o `generateSecret-app.py` for executado novamente obtêm-se segredos com todos os caracteres possíveis:

```
arrow@arrowVM:~/Cripto-py-develop$ python3 generateSecret-app.py 1234
Wb0RtdF°Üü{p0I]æ
6LL:_%B\@M
      cÍÜyÄ&PÊÜ/iHæÄËUdİð
µ ç%3ó1;g[*0 i¥iKÖx¬;İ\r50cE"%²m/é?"@p³3%ãÖzSèúİ%0/0Aa01nhÊHéúRñä÷Ý
yLU%3ðñk*ÊÊ7éÊð&ıİ 2'²N[εı r
ÊÂ89öLúsnÊ9âpø±³İV♦N _/'^c²(g(BHµãð♦DSWçËY$ı;PÑúKð"50ý~ñÜikİp4_ÜLÍ°°0
!ÜBİier;ÊLÊøt0$WeÊ·"nyërpe}8ÂUısxç0ø×ıSI¶"UóæUÁyy7BµİÜ"nô"nBÜİÜHÁ
3°"as İµøLLV
/0dCÿ9ø0¥QB0x÷K×3Xİİ:¶q «6UoİÊX±ÿ²Öá«[0+øx± SàW1
ÿ,n,ÿAH¥é;_İxék =Āy(sþÄÉ·øDGsyÄz('♦Ü;CZ{ >^ÊşzËxĀ,
      İð-g d
ø0^G,_*ô5¶ð,1Æ
?ýqð÷|4U Öâ|c S¥
İ*2¶~TZp%{:C=HNP%{+': MRúÜj²9¶p²?ş#òb4p0áyÍy%,0+ô¹+nALL"qfémçÑç/7áyİ
rÖ³0G*ß)~İT.CmâD)İ|ðNöká'°%Ö~ç3ðÊ?Qc,"*-óèUJ0İfd&
      İ %KH
C]zDu·İâú2ıLÊÜñ;Nİck("¥c,#éWAËupZİó[°¹0Üçqİz Ö(ø»0 Nð=d,Àçİró~pYey>İ6¥Ú]~ÄÄ÷
ø³bİİ'ä×QM)ôµxalÊHNP$ \SHÁñÜDÜ]Ê
```

Figura 1.4: Novo *output* gerado pelo ficheiro `generateSecret-app.py`

Por fim, executou-se o programa inserido no ficheiro `recoverSecretFromComponents-app.py` com inserção de 3, 4 e 2 componentes:

Figura 1.8: Execução do código do ficheiro `recoverSecretFromComponents-app.py` com inserção de 3 componentes

Figura 1.9: Execução do código do ficheiro `recoverSecretFromComponents-app.py` com inserção de 4 componentes

Figura 1.10: Execução do código do ficheiro `recoverSecretFromComponents-app.py` com inserção de 2 componentes

B. Indique também qual a diferença entre `recoverSecretFromComponents-app.py` e `recoverSecretFromAllComponents-app.py`, e em que situações poderá ser necessário utilizar `recoverSecretFromAllComponents-app.py` em vez de `verSecretFromComponents-app.py`.

Será necessário utilizar todos os componentes em vez do número mínimo quando, por exemplo, o número de componentes necessárias para aceder ao segredo equivale ao número total de componentes, isto é, quando se tem um esquema simples de divisão/partilha de segredos.

1.3 Pergunta P.IX.4.1

O certificado da autoridade de certificação pedido foi o de França, para a EC "Idemia Identity & Security France".

¹A solução da pergunta 3.1 encontra-se na pasta P-IX-3.1 no git do grupo de trabalho.

O resultado do comando «openssl x509 -in cert.crt -text -noout» está dividido nas três imagens seguintes.

```
Ficheiro  Editar  Ver  Favoritos  Configuração  Ajuda
luis@Luis-VivoBook:~$ openssl x509 -in cert.crt -text -noout
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      11:20:be:df:3d:d2:a0:2e:9b:4a:9f:24:b9:ca:ae:15:7c:43
    Signature Algorithm: sha512WithRSAEncryption
    Issuer: C = FR, O = Idemia Identity & Security France, organizationIdentifier = N
    TRFR-440305282, CN = IDEMIA eIDAS Root CA
    Validity
      Not Before: Jun 30 00:00:00 2020 GMT
      Not After : Jun 30 00:00:00 2030 GMT
    Subject: C = FR, O = Idemia Identity & Security France, organizationIdentifier =
    NTRFR-440305282, CN = IDEMIA eIDAS Qualified CA
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public-Key: (4096 bit)
      Modulus:
        00:ac:7a:46:35:bc:92:51:6a:25:0d:f9:08:9e:fa:
        51:cf:54:b3:06:2a:9f:79:c1:9d:3a:4d:97:93:94:
        11:5c:9e:26:5b:8c:a3:82:46:64:07:38:6e:f6:8f:
        dd:21:2a:0b:f3:d9:49:34:97:2e:7e:21:a8:1d:c5:
        53:18:a1:c3:38:9d:59:53:84:2a:e0:19:06:99:56:
        30:30:e1:38:b5:bf:a6:87:35:0b:83:05:2a:a4:e6:
        78:40:5a:7e:36:d0:27:95:cc:bf:23:df:45:3c:4c:
        ae:ea:70:de:7e:3e:c2:c5:51:a8:11:54:2c:f6:bc:
        49:2a:e7:69:2b:9b:2f:1e:f7:d9:9d:51:f7:b5:56:
        d0:18:68:51:75:7e:3d:15:19:6a:f4:02:57:5b:45:
        fb:da:8c:ca:89:55:f1:c3:bc:bf:8e:f8:7d:26:ef:
        26:ab:06:c1:78:3c:42:d9:35:07:6b:29:22:e9:65:
        f0:9b:d3:69:7a:6a:51:25:46:05:d5:b5:35:2e:2c:
        8a:0e:13:7b:51:d0:77:87:9f:ab:52:5d:41:3c:ba:
        8b:70:72:5e:d5:16:b0:ad:be:fa:3d:b8:96:95:53:
        45:1b:7d:38:bd:2c:32:a0:3a:49:d5:3e:00:29:0b:
        16:98:61:74:bb:4f:ab:9c:8b:b2:e7:d0:d6:0d:53:
        67:3f:02:c1:87:28:62:20:a0:12:27:ba:d6:7a:9e:
        c0:f2:9d:3c:ae:9d:34:99:5e:d1:c9:11:e2:50:f3:
        7e:45:2e:89:7a:40:9a:9e:76:e3:28:16:fb:a0:f8:
        e8:fd:84:86:22:01:2d:99:b2:c7:5d:51:5b:50:93:
        28:84:77:25:36:ea:9c:0e:eb:99:42:3e:d6:08:4d:
```

Figura 1.11: Conteúdo do certificado para a entidade de certificação pedida (1)

```
Ficheiro  Editar  Ver  Favoritos  Configuração  Ajuda
5a:70:cd:4d:ba:0f:98:c9:ab:1b:25:02:c3:c6:4d:
4f:f1:87:05:59:55:00:3a:2f:d5:a6:16:59:bd:93:
f0:35:e1:f5:60:f5:10:7f:93:d8:8c:28:d8:68:dc:
cd:bc:dd:a7:de:69:ba:d6:3f:dd:00:76:da:00:82:
4e:7f:ea:cf:ba:15:1b:b7:7d:8f:80:30:14:4d:45:
e6:14:cb:60:cc:76:c3:d8:64:3d:5e:1f:eb:b5:4a:
2c:4f:98:72:cb:5d:cb:bf:d8:43:45:70:da:3c:5c:
bf:33:eb:3e:71:73:ed:b3:3e:2e:43:06:c5:da:cf:
a7:f4:19:28:e9:4b:af:4a:9b:f3:f5:5f:db:e6:af:
88:e0:86:14:c6:66:1f:d4:6e:d1:b7:30:1d:f3:83:
60:c9:e1:69:ea:d1:6c:64:d7:00:52:66:a2:40:23:
d6:2d:41:87:25:1c:89:11:7a:9a:bc:7f:78:ea:4b:
06:ce:1d
Exponent: 65537 (0x10001)
X509v3 extensions:
  X509v3 Key Usage: critical
    Certificate Sign, CRL Sign
  X509v3 Certificate Policies:
    Policy: X509v3 Any Policy
    CPS: http://pki.trust.idemia.io/policies/

  X509v3 Basic Constraints: critical
    CA:TRUE, pathlen:0
  X509v3 CRL Distribution Points:

Full Name:
  URI:http://pki.trust.idemia.io/crl/idemia-eidas-root-ca.crl

Authority Information Access:
  CA Issuers - URI:http://pki.trust.idemia.io/cer/idemia-eidas-root-ca.cer

X509v3 Subject Key Identifier:
  51:0C:F0:0F:43:84:9C:CA:4F:7F:CF:F3:3B:B7:8C:42:7D:A9:F3:BD
X509v3 Authority Key Identifier:
  keyid:33:FA:B6:BF:39:12:4C:E9:1A:E0:3E:DA:93:AB:24:12:F5:A5:14:08

Signature Algorithm: sha512WithRSAEncryption
30:06:c6:26:81:0f:34:e6:4c:3c:14:90:b7:89:c0:3c:d3:3c:
0e:4f:7a:0f:db:db:11:df:cd:7b:23:94:a5:29:7b:16:0c:d6:
f6:eb:68:ac:1e:31:3a:3c:dc:14:e6:aa:8a:23:b3:6b:4a:93:
```

Figura 1.12: Conteúdo do certificado para a entidade de certificação pedida (2)

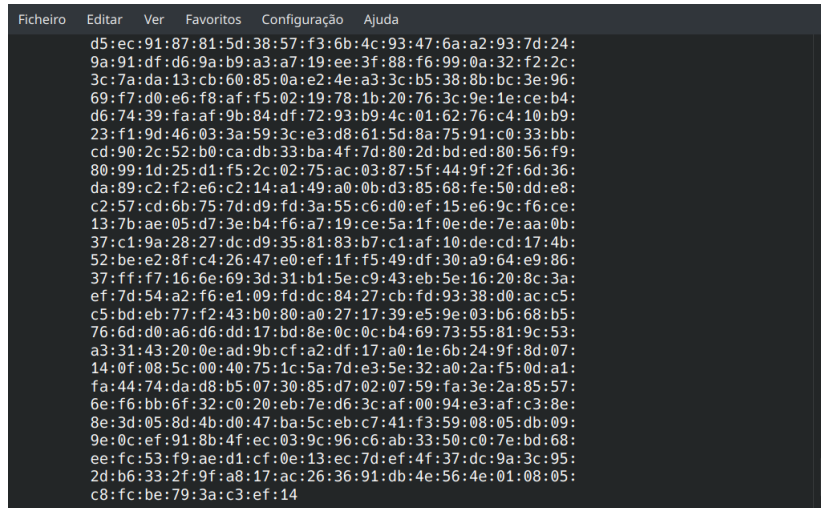


Figura 1.13: Conteúdo do certificado para a entidade de certificação pedida (3)

O algoritmo usado é o SHA-512 com cifra RSA, cuja chave pública tem 4096 bits. De seguida, é apresentada uma tabela que contém o estado de aprovação para diversos algoritmos usados para gerar e verificar assinaturas digitais.

Digital Signature Process	Domain Parameters	Status
Digital Signature Generation	< 112 bits of security strength: DSA: $(L, N) \neq (2048, 224), (2048, 256)$ or $(3072, 256)$ ECDSA: $\text{len}(n) < 224$ RSA: $\text{len}(n) < 2048$	Disallowed
	≥ 112 bits of security strength: DSA: $(L, N) = (2048, 224), (2048, 256)$ or $(3072, 256)$ ECDSA or EdDSA: $\text{len}(n) \geq 224$ RSA: $\text{len}(n) \geq 2048$	Acceptable
Digital Signature Verification	< 112 bits of security strength: DSA ³² : $((512 \leq L < 2048)$ or $(160 \leq N < 224))$ ECDSA: $160 \leq \text{len}(n) < 224$ RSA: $1024 \leq \text{len}(n) < 2048$	Legacy use
	≥ 112 bits of security strength: DSA: $(L, N) = (2048, 224), (2048, 256)$ or $(3072, 256)$ ECDSA and EdDSA: $\text{len}(n) \geq 224$ RSA: $\text{len}(n) \geq 2048$	Acceptable

Figura 1.14: Estado de aprovação dos algoritmos usados para gerar e verificar assinaturas digitais

É aceite o uso de uma chave RSA cujo tamanho do módulo seja maior ou igual a 2048. Como o tamanho da chave utilizado pela entidade de certificação é o dobro, 4096, considera-se que é adequado e que, de momento, não são precisas melhorias para melhorar a sua segurança.

O mesmo aplica-se para o hash utilizado (SHA-512).

Security Strength	Digital Signatures and Other Applications Requiring Collision Resistance	HMAC, ⁷⁰ KMAC, ⁷¹ Key Derivation Functions, ⁷² Random Bit Generation ⁷³
≤ 80	SHA-1 ⁷⁴	
112	SHA-224, SHA-512/224, SHA3-224	
128	SHA-256, SHA-512/256, SHA3-256	SHA-1, KMAC128
192	SHA-384, SHA3-384	SHA-224, SHA-512/224, SHA3-224
≥ 256	SHA-512, SHA3-512	SHA-256, SHA-512/256, SHA-384, SHA-512, SHA3-256, SHA3-384, SHA3-512, KMAC256

Figura 1.15: Segurança máxima fornecida para o hash e funções à base de hash

1.4 Pergunta P.IX.5.1

Pretende-se que altere o código fornecido para a experiência 5.2, de forma a simplificar o input e output.²

Assinante

Para a inicialização dos componentes, alterou-se o *output* do programa de modo a que fosse possível separar o *return* do R' e das componentes inicializadas.

A execução do ficheiro `init-app.py` com a opção `-init` simplesmente retorna o R':

```
arrow@arrowVM:~/Cripto-py-develop$ python3 init-app.py -init
pRDashComponents: 39b49a6faf7c30c489b63a0b2632fb34a4878ee18d4ef11e8a0663541066ab
84.f8b521050af444992cadf170abfcfb94cbf5006e412b21aaea2e0ed03ce80c9e
arrow@arrowVM:~/Cripto-py-develop$
```

Figura 1.16: Execução com o argumento `-init` do ficheiro `init-app.py`

Sem a opção `-init`, o programa inicializa todas as componentes e guarda-as num ficheiro "signerFile":

```
arrow@arrowVM:~/Cripto-py-develop$ python3 init-app.py
Components initialized in "signerFile"
arrow@arrowVM:~/Cripto-py-develop$ cat signerFile
-----Initialized Components-----

1dc92dd5a8ba4fba885070fb62c93e6b42ea1cc2351604eb95c62440b8c2f26f.1eb032603277519
acb72bc57eacfa94604c87b6d6afd9cf44055cc126ceda7b1

-----R'-----

1dc92dd5a8ba4fba885070fb62c93e6b42ea1cc2351604eb95c62440b8c2f26f.63e9faf4b217f94
f718128325e78a02c48491debd6ea8fdec3232e836be51593
```

Figura 1.17: Execução sem argumentos do programa `init-app.py` e respetivo resultado

Para criar a assinatura, gerou-se uma nova chave privada com base na experiência 5.1:

²Para análise mais detalhada do código alterado: consultar a Pasta P-IX-5.1 do github do grupo de trabalho.


```

arrow@arrowVM:~/Cripto-py-develop$ openssl ecparam -name prime256v1 -genkey -noout -out pkey.pem
arrow@arrowVM:~/Cripto-py-develop$ openssl req -key pkey.pem -new -x509 -days 365 -out pkey.crt
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]:PT
State or Province Name (full name) [Some-State]:Braga
Locality Name (eg, city) []:Braga
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Uminho
Organizational Unit Name (eg, section) []:Uminho
Common Name (e.g. server FQDN or YOUR name) []:Ariana Lousada
Email Address []:pg47034@alunos.uminho.pt
arrow@arrowVM:~/Cripto-py-develop$

```

Figura 1.18: Criação de uma nova chave e respetivo certificado com o *openssl*

Alterando então o código do programa como pedido no enunciado e utilizando uma *blind message* (criada com o programa *ofusca-app.py*, figura 1.20) e chave criadas anteriormente:

```

arrow@arrowVM:~/Cripto-py-develop$ python3 blindSignature-app.py -key pkey.pem -bmsg
1819ec07c2a714d147dd1054580706d2912c9ba06239c71adbdc8ee4f472efa
Passphrase: helloarrow

Blind signature: 35bc69be45a96c168f2a45ef679e65abd0b69768a4d0057989307e353eec9ece998
1f607711b3cfaf841db500fba59b583f244aa94e831cd2f7f88461c7588
arrow@arrowVM:~/Cripto-py-develop$

```

Figura 1.19: Criação de uma assinatura cega com o programa *blindSignature-app.py*

O programa retira os *InitComponentes* do ficheiro do assinante "signerFile" onde foram armazenados anteriormente.

Requerente

Para ofuscar uma determinada mensagem, alterou-se o programa `generateBlindData-app.py` fornecido pela equipa docente de modo a este retornar a *blind message* guardando os restantes componentes em ficheiro:

```
arrow@arrowVM:~/Cripto-py-develo$ python3 ofusca-app.py -msg "Hello, how are you?"  
-RDash b02d5f5a26f750e6cc2ae7bd0d3fc469a9a22329e447542d319d5564d7ef2960.e6c2fa6d3bc3  
5a57a17d50f3960ac9c80f4b5855f77a596e01461e53ec509ee6  
  
Blind Message: 1819ec07c2a714d147dd1054580706d2912c9ba06239c71adbdc8ee4f472efa  
  
Other components saved in "BlindDataFile"  
  
arrow@arrowVM:~/Cripto-py-develo$ cat BlindDataFile  
  
-----Blind Components-----  
  
2083d038f0c5727a0a67418c915a24a602091349a74dc9a83a33077b617a4e80.43ba378373005bcf758  
ac46d1faae6adca83de01f833ed837b0acc4e2803802e  
  
-----pRComponents-----  
  
43ba378373005bcf758ac46d1faae6adca83de01f833ed837b0acc4e2803802e.773fff8cadf5e43fa08  
d82c334e337476376baee129da3e48cc77b7e21383de0
```

Figura 1.20: Ofuscação de uma mensagem utilizando o programa `ofusca-app.py`

Para desofuscar uma determinada assinatura, são pedidos os `pRComponents` juntamente com a assinatura cega. O programa retira os *blindComponents* do ficheiro `BlindDataFile` no qual foram armazenados anteriormente:

```
arrow@arrowVM:~/Cripto-py-develo$ python3 desofusca-app.py -s 35bc69be45a96c168f2a4  
5ef679e65abd0b69768a4d0057989307e353eec9ece9981f607711b3cfaf841db500fba59b583f244aa9  
4e831cd2f7f88461c7588 -RDash 43ba378373005bcf758ac46d1faae6adca83de01f833ed837b0acc4  
e2803802e.773fff8cadf5e43fa08d82c334e337476376baee129da3e48cc77b7e21383de0  
  
Signature: 650f18b4aab8d91fa652125539e95707bcb10e6d5e7fb05825b798ba2e437aef
```

Figura 1.21: Desofuscação de uma assinatura utilizando o programa `desofusca-app.py`

Verificador

De modo a ser possível verificar a assinatura, utiliza-se o `openssl` para retirar a chave pública do certificado `.crt` gerado na primeira fase da pergunta.

Inserindo a mensagem inicial em `plaintext` e a assinatura obtida com o Requerente:

```
arrow@arrowVM:~/Cripto-py-develo$ python3 verify-app.py -cert pkey.crt -msg "Hello,  
how are you?" -sDash 650f18b4aab8d91fa652125539e95707bcb10e6d5e7fb05825b798ba2e437a  
ef -f BlindDataFile  
Invalid signature
```

Figura 1.22: Verificação da assinatura sobre a mensagem inicialmente ofuscada com o programa `verify-app.py`

Apesar de executar os passos ordenadamente, a equipa de trabalho não conseguiu obter uma assinatura válida para a mensagem inserida.

1.5 Pergunta P.IX.6.1

Foi contratado para ajudar uma conhecida empresa de análises a guardar todos os dados das análises dos seus clientes em ambiente cloud. O que a empresa pretende guardar é um ficheiro com a seguinte estrutura por cada linha: NIC do cliente, seguido de uma lista de tuplos (tipo de análise, valor). Por exemplo, uma linha pode ser: "123456789, (A23, 12,2), (B4, 32,1), (A2, 102), (CAA2, 34,5)". Adicionalmente foi-lhe indicado que poderá ser necessário obter a média de uma ou mais tipos de análise.

1. A melhor maneira para guardar esta informação em ambiente cloud seria cifrando a informação através de *Partially homomorphic encryption* e guarda-la numa base de dados.

2. A melhor maneira de efectuar cálculos em ambiente de cloud sem estes serem decifrados é utilizando *Partially homomorphic encryption*. Este método de cifra permite que seja possível efectuar operações aritméticas sobre dados cifrados, com isto podemos calcular a média de uma ou mais tipos de análises.

Isto é especialmente importante para informação sensível e com políticas de privacidade restritas, como neste caso, análises de pacientes. Isto permite que mesmo que a informação seja comprometida pela entidade que efectua operações aritméticas a informação continua segura.

3. Aqui está uma possível maneira para a empresa testar esta solução localmente. Foi desenvolvida em Python juntamente com a livreria PHE, que nos permite cifrar dados usando criptografia homomórfica.

Em primeiro lugar, importou-se a livreria PHE.

```
1 import phe
2 from phe import paillier
```

Figura 1.23: Importação da livreria PHE

Em segundo lugar, chamou-se a função 'GerarKeys' para criar uma chave pública e outra privada para a cifração dos dados.

```
7 def GerarKeys():
8     pub_key, priv_key = paillier.generate_paillier_keypair() ## Gera a chave publica e privada
```

Figura 1.24: Gerar key privada e publica

Foi criada também uma função chamada 'InserirAnalises' que permite a inserção de várias análises de um paciente identificado através do NIC num ficheiro de texto.

```
20 def InserirAnalises(NIC, Analises): #Insere analises de um paciente no ficheiro
21     #Exemplo
22     #"123456789, (A23, 12,2), (B4, 32,1), (A2, 102), (CAA2, 34,5)"
23     Linha = str(NIC) + ', '
24     for analise in Analises:
25         Linha = Linha + str(analise)
26         if(type(analise) != str):
27             Linha = Linha + ','
28         else:
29             Linha = Linha + ', '
30
31     l = len(Linha)
32     Linha = Linha[:l-1]
33     Linha = Linha + '\n'
34
35     fp = open('analises.txt', 'a')
36     fp.write(Linha)
37     fp.close()
```

Figura 1.25: Função para adicionar analises

Para este teste foram inseridos dois clientes que fizeram várias análises, um cliente com o NIC 987654321 e outro com o NIC 123456789.

```

80 Analises = ["(A23", 12.3, "(B4", 32.5, "CAA2", 34,5]
81 InserirAnalises(987654321, Analises)
82
83 Analises = ["(C23", 12, "(Z2", 132.7]
84 InserirAnalises(123456789, Analises)
85
86 Analises = ["(Y13", 12.1, "(B4", 16.6, "(K76", 78.13]
87 InserirAnalises(987654321, Analises)

```

Figura 1.26: Análises adicionadas para o teste

Este é o conteúdo do ficheiro 'analises.txt' resultante após a adição destas análises

```

1 987654321, (A23, 12.3),(B4, 32.5),CAA2, 34),5)
2 123456789, (C23, 12),(Z2, 132.7)
3 987654321, (Y13, 12.1),(B4, 16.6),(K76, 78.13)
4

```

Figura 1.27: Ficheiro analises.txt

Foi também criada uma função responsável por enviar os dados cifrados para a cloud, para calcular a média de uma análise específica de um cliente identificado através do NIC.

```

40 def CalcularMedia(NIC, Analise): #Calcula a média de determinado valor de analise
41     file = open('analises.txt', 'r')
42     Lines = file.readlines()
43
44     valoresCifrados = []
45
46     for line in Lines:
47         currentNIC = line.split(",")
48         if(currentNIC[0] == str(NIC)):
49             valoresCifrados.append(CifrarValor(float(GetValue(line, Analise)), pub_key))
50
51     media = CloudCalcularMedia(valoresCifrados) #Envia para a cloud os valores já cifrados
52
53     return(priv_key.decrypt(media))
54

```

Figura 1.28: Função responsável de enviar para a cloud a informação cifrada

Nesta função o ficheiro analises.txt é aberto e percorre todas as linhas, verificando se o NIC corresponde ao cliente. Caso corresponda, verifica se tem a análise pretendida através da função 'GetValue', este valor é cifrado com a chave pública e adicionado à lista 'valoresCifrados'. Esta lista de valores cifrados é enviada para a cloud onde esta calcula a média sem decifrar os valores e retorna o resultado. Por último a função 'CalcularMedia' devolve a média pedida decifrada.

Função 'GetValue':

```

56 def GetValue(line, Analise):
57     splitted = line.split(',')
58     trigger=False
59     valor = ''
60     for x in splitted:
61         if(trigger):
62             valor = "".join(_ for _ in x if _ in ".1234567890")
63             break
64         if(Analise in x):
65             trigger=True
66
67     if(valor==''):
68         return 0
69     return valor

```

Figura 1.29: Função GetValue

Função para calcular a média na cloud com os valores cifrados e que devolve o resultado também cifrado:

```

72 def CloudCalcularMedia(valores): #Parte calculada na cloud com valores cifrados
73
74     media = sum(valores) / len(valores)
75     return media
76

```

Figura 1.30: Função que corre na cloud

Aqui vê-se o resultado da média da análise B4 para o paciente com o NIC 987654321.

```

91 print("Media das analises B4 do paciente com NIC 987654321 é: " + str(CalcularMedia(987654321, 'B4')))
92
Media das analises B4 do paciente com NIC 987654321 é: 24.55
[Finished in 2.9s]

```

Figura 1.31: Resultado da media

Capítulo 2

Referências

- <https://pycryptodome.readthedocs.io/en/latest/src/cipher/chacha20.html>
- <https://esignature.ec.europa.eu/efda/tl-browser/#/screen/tl/FR/29/1>
- NIST Special Publication 800-131A Revision 2 - Transitioning the Use of Cryptographic Algorithms and Key Lengths (Mar 2019), <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-131Ar2.pdf>
- NIST Special Publication 800-57 Part 1 Revision 5 - Recommendation for Key Management: Part 1 – General (Maio 2020), <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf>
- <https://pypi.org/project/pyOpenSSL/>