



UNIVERSIDADE DO MINHO

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA
ENGENHARIA DE SEGURANÇA

Aula2

GRUPO 4

Autores

JOÃO SOUSA	(A77768)
FRANCISCO ARAÚJO	(A79281)

19 de Fevereiro de 2019

Conteúdo

1	Números aleatórios/pseudoaleatórios	2
1.1	Pergunta 1	2
1.2	Pergunta 2	2
1.3	Pergunta 3	2
1.3.1	Pergunta 3.1	2
1.3.2	Pergunta 3.2	2
2	Partilha/Divisão de segredo	3
2.1	Pergunta 2.1-A	3
2.2	Pergunta 2.1-B	3
3	Authenticated Encryption	3
4	Algoritmos e tamanhos de chaves	4

1 Números aleatórios/pseudoaleatórios

1.1 Pergunta 1

As conclusões que podem ser tiradas depois de executados os quatros comandos apresentados na ficha prática, é que a geração de números pseudoaleatórios até 64 bytes, o sistema é extremamente rápido a processar esses bytes pois como o tamanho não é muito grande, o sistema possui entropia suficiente para responder de imediato, uma vez que com a execução do comando com especificação *random* tem em conta a entropia do sistema para gerar os tais números pseudoaleatórios.

Pelo contrário, o mesmo não se verificou quando tentamos gerar números pseudoaleatórios de 1024 bytes, pois neste caso o sistema não possui entropia suficiente para os gerar, demorando algum tempo a ser processado o respetivo comando, para o caso em que o comando contenha a especificação *random*.

Caso a especificação seja *urandom*, neste caso o sistema gera um número pseudoaleatório sem ter em conta a entropia do sistema, sendo o resultado imediato.

1.2 Pergunta 2

Neste caso após a instalação do software *haveged*, constatamos que para gerar um número pseudoaleatório de tamanho 1024 bytes foi imediato tanto para o comando com a especificação *random* como para o comando com a especificação *urandom*, o mesmo se sucedeu para os de tamanho inferior, que já acontecia anteriormente.

Isto acontece, pois o *haveged* é um software bastante usado que tem como objetivo explorar a volatilidade do estado do hardware como fonte de incerteza, o que leva a que a entropia no sistema seja bastante mais elevada do que acontecia quando este software não se encontrava ativo.

1.3 Pergunta 3

1.3.1 Pergunta 3.1

O output contém apenas números e letras, uma vez que a função usada no respetivo código, denominada *generateSecret* importada do package *eVotUM/Cripto/shamirsecret* recebe unicamente o tamanho do número pretendido, e nessa função é apenas utilizado os caracteres que se encontram no *string.ascii_letters* e no *string.digits*, sendo assim, o uso de caracteres especiais inexistentes.

1.3.2 Pergunta 3.2

Para usar caracteres especiais, a melhor forma seria modificar a função *generateSecret*, mais concretamente na linha 256 do ficheiro *eVotUM/Cripto/shamirsecret.py* de modo a permitir a utilização de todo o tipo de caracteres, podendo também colocar os caracteres que o utilizador entender.

```
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:
            if (c in (string.ascii_letters + string.digits) and l < secretLength): # printable character
                l += 1
                secret += c
    return secret
```

Figura 1: *Função que deve ser alterada*

2 Partilha/Divisão de segredo

2.1 Pergunta 2.1-A

Para dividir o segredo presente no enunciado recorreremos ao comando *python createSharedSecret-app.py number_of_shares quorum uid private-key.pem* onde o *number_of_shares* foi oito e o *quorum* foi 5 e o *uid* foi g4 identificando assim o grupo. Antes disso foi necessário criar uma chave privada recorrendo ao seguinte comando *openssl genrsa -aes128 -out mykey.pem 1024*.

Depois de executado tudo isto, obtivemos os 8 componentes correspondentes a cada parte.

2.2 Pergunta 2.1-B

O ficheiro *recoverSecretFromComponents-app.py* tem como objetivo recuperar apenas os componentes necessários para a recuperação do segredo, isto é, o número de *quorum*.

Pelo contrário o ficheiro *recoverSecretFromAllComponents-app.py* tem como objetivo recuperar todas as componentes em quais o segredo foi dividido. Este último ficheiro pode ser utilizado por exemplo, quando existe a necessidade de alterar o segredo, sendo preciso estarem os oito componentes para haver a recuperação e a respetiva distribuição das novas componentes.

3 Authenticated Encryption

A empresa deveria implementar o *Encrypt-then-MAC (EtM)*, isto é primeiro é realizada a cifra do texto e depois é realizado o *MAC* sobre o texto cifrado. Depois disso o texto cifrado e o *MAC* são enviados em conjunto. Este tipo de *Authenticated Encryption* é melhor do que o *Encrypt-and-MAC (E&M)* e *MAC-then-Encrypt (MtE)*.

Para ajudar a entender o algoritmo que deveria ser implementado foi realizado pseudo-código:

```

cifrar(plaintext):
    textocifrado = cifra(plaintext)
    mensagem_com_data = cyphertext+data
    hmac = (key, mensagem_com_data)
    return mensagem_com_data+hmac

decifrar(textocifrado):
    Hmac = retirarHmac(textocifrado)
    h=verificarMac(Hmac,key)
    if (h==True):
        data=getData(textocifrado)
        textocifrado_sem_data = retirarData(textocifrado)
        if (anuidade esta paga no dia da data):
            return decifrar(textocifrado,key)
        else:
            return Null
    else:
        return Null

```

4 Algoritmos e tamanhos de chaves

Nesta secção era-nos pedido para dado um país, no caso do nosso grupo, a Hungria, identificar o tamanho das chaves e os algoritmos presentes nos certificados *QCert for ESig*, emitidos pelas Entidades de Certificação (EC) que emitem certificados digitais qualificados.

Posto isto, optamos por escolher as seguintes Entidades de certificação:

- **Microsec Micro Software Engineering Consulting Private Company Limited by Shares**

Como esta EC já tinha emitido vários certificados, optamos pelo certificado emitido mais recentemente como nos era pedido, denominado *Qualified KET e-Szigno CA 2018*, que data do dia 31-12-2018. De seguida, realizamos os passos estipulados no enunciado, e obtivemos o seguinte:

```

Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      bc3d3d9a56d441a2bb5987020a
    Signature Algorithm: sha256WithRSAEncryption
    Issuer: C=HU, L=Budapest, O=Microsec Ltd., CN=Microsec e-Szigno Root CA 2009/emailAddress=info@e-szigno.hu
    Validity
      Not Before: Sep  6 09:00:00 2018 GMT
      Not After:  Dec 29 09:00:00 2029 GMT
    Subject: C=HU, L=Budapest, O=Microsec Ltd./2.5.4.97=VATHU-23584497, CN=Qualified KET e-Szigno CA 2018
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      Public-Key: (2048 bit)
      Modulus:
        00:ec:52:2e:87:2a:20:06:90:50:b6:04:02:fe:cc:
        1f:c4:1b:ff:95:ca:70:ab:db:f4:15:c6:51:68:b4:
        26:6d:20:c5:d4:0c:58:72:1c:fc:ef:3e:ba:cf:76:
        c0:51:93:1c:76:17:84:4b:24:fd:fc:6c:ab:d4:8a:
        84:39:20:74:1f:2b:03:91:25:fb:30:02:42:52:6a:
        4d:77:ef:eb:55:d6:3a:88:f3:0c:1e:55:aa:16:f1:
        02:bc:cb:9c:c7:50:b9:f1:07:f0:32:7c:56:9a:ea:
        22:46:96:cb:b3:74:aa:43:dd:31:c3:77:c9:78:ce:
        42:7d:e2:3c:f2:e4:b9:cd:fe:fa:9b:f9:a5:58:00:
        99:51:50:12:06:bb:f1:ad:51:f3:c5:dd:f2:82:da:
        11:2b:1a:a2:f5:da:4a:9e:c8:d4:cf:d6:4a:c1:48:
        1f:8d:15:b0:1c:e0:dd:78:07:08:32:8e:f6:87:97:
        02:49:0a:91:32:13:0b:56:27:ce:1d:83:16:cf:d7:
        9b:38:aa:39:08:16:89:65:3b:1e:9e:24:bb:46:bb:
        e3:64:58:a9:05:0c:29:ba:a8:5e:12:53:48:3d:5f:
        a1:a5:4e:7f:51:2d:b4:73:fd:c4:d5:7c:4c:ac:97:
        e7:d6:d8:c3:d8:b4:66:d2:38:cd:6e:7c:8f:c7:9c:
        49:f3
      Exponent: 65537 (0x10001)

```

Figura 2: Informação relativa ao respetivo certificado

```

X509v3 extensions:
  X509v3 Basic Constraints: critical
    CA:TRUE
  X509v3 Key Usage: critical
    Certificate Sign, CRL Sign
  X509v3 Certificate Policies:
    Policy: X509v3 Any Policy
    CPS: http://cp.e-szigno.hu/acps

  X509v3 Subject Key Identifier:
    55:46:0C:F9:D3:99:A3:D4:6C:06:CB:FD:78:DE:47:E7:7B:3E:1E:95
  X509v3 Authority Key Identifier:
    keyId:CB:0F:C6:DF:42:43:CC:3D:CB:B5:48:23:A1:1A:7A:A6:2A:BB:34:68

  X509v3 CRL Distribution Points:

    Full Name:
      URI:http://rootca2009-crl1.e-szigno.hu/rootca2009.crl

    Full Name:
      URI:http://rootca2009-crl2.e-szigno.hu/rootca2009.crl

    Full Name:
      URI:http://rootca2009-crl3.e-szigno.hu/rootca2009.crl

  Authority Information Access:
    OCSP - URI:http://rootca2009-ocsp1.e-szigno.hu
    OCSP - URI:http://rootca2009-ocsp2.e-szigno.hu
    OCSP - URI:http://rootca2009-ocsp3.e-szigno.hu
    CA Issuers - URI:http://rootca2009-ca1.e-szigno.hu/rootca2009.crt
    CA Issuers - URI:http://rootca2009-ca2.e-szigno.hu/rootca2009.crt
    CA Issuers - URI:http://rootca2009-ca3.e-szigno.hu/rootca2009.crt

  Signature Algorithm: sha256WithRSAEncryption
    ac:30:a5:e3:f1:ad:7e:f6:0f:80:b2:38:6a:d3:27:6b:9b:c1:
    68:8c:29:88:ao:a5:7a:94:af:33:38:83:b1:a7:54:d5:7a:5f:
    d3:cf:ba:aa:11:6a:bf:19:e4:30:d4:14:d1:d4:f8:f6:c7:4d:
    c4:e8:67:25:52:f0:38:2d:c3:47:45:de:51:52:75:10:a6:03:
    14:91:ee:3b:95:a1:a6:25:00:fd:51:88:f9:28:e5:e2:78:80:
    62:8e:11:60:32:bd:c4:0c:be:8a:32:5b:c3:cc:bb:84:d5:
    c6:23:0c:0c:90:72:21:c2:d3:95:f1:a6:6a:bc:0b:48:59:32:
    d8:65:ec:25:0d:5d:c6:fb:45:ed:c6:57:fb:c4:1e:d1:11:61:
    d2:45:b3:e6:30:e2:77:e6:3b:4d:9e:6a:e8:45:e8:cd:90:18:
    e4:da:fc:e0:e2:a2:bb:9e:b7:2f:5f:51:ea:fd:e0:bf:61:da:
    26:c5:03:30:57:e8:6f:3f:b9:71:d1:ac:78:74:e1:d3:ec:d8:
    43:25:28:59:97:3e:84:21:17:2f:1e:82:0a:93:d2:32:61:c2:
    38:08:95:85:19:b3:d5:8a:74:33:78:8e:de:10:7f:42:56:9b:
    c6:80:af:c4:51:28:6f:7b:d0:aa:93:69:c4:de:1b:e9:05:9d:
    85:9a:5e:8e

```

Figura 3: Informação relativa ao respetivo certificado

Dado o que encontramos, podemos concluir que o algoritmo utilizado é o *RSASignature*, e o tamanho da chave é de 2048 bit. Existem também outras informações relevantes como o algoritmo de assinatura utilizado.

- NISZ National Infocommunications Services Company Limited by Shares

Para esta EC tal como na primeira, escolhemos o emitido mais recentemente, denominado *Minősített Közigazgatási Tanúsítványkiadó - GOV CA*. Refizemos os mesmos passos que tínhamos feito para o anterior certificado e descobrimos que o algoritmo de assinatura e o tamanho da chave são iguais ao anterior.

```

-----BEGIN-----
Version: 3 (0x02)
Serial Number: 11228913 (0x43c09a1)
Signature Algorithm: sha256WithRSAEncryption
Issuer: CN=, OU=KÖZMŰK (Public Administration Root CA - Hungary)
Validity
Not Before: Aug 28 00:00:00 2014 GMT
Not After:  Aug 28 00:00:00 2016 GMT
Subject: CN=, OU=KÖZMŰK, postalCode=1052, serialNumber=11228913, email=info@hiteles.gov.hu, cn=Közigazgatási Tanúsítványkiadó
c3f0a3
-----END-----
Public Key Info:
Public Key Algorithm: rsaEncryption
Public Key: (2048 bit)
Modulus:
00:9a:00:9a:ff:2c:7d:04:04:12:00:a2:0d:c3:00:
70:70:70:70:70:70:70:70:70:70:70:70:70:70:70:70:
13:00:00:17:00:00:17:00:00:17:00:00:17:00:00:17:00:
8a:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:
8a:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:
93:02:02:04:00:00:00:00:00:00:00:00:00:00:00:00:00:
70:70:70:70:70:70:70:70:70:70:70:70:70:70:70:70:
8a:02:04:12:00:a2:0d:c3:00:70:70:70:70:70:70:70:70:
a7:00:70:70:70:70:70:70:70:70:70:70:70:70:70:70:
69:c7:01:70:00:12:00:a2:0d:c3:00:70:70:70:70:70:70:
8a:00:00:70:70:70:70:70:70:70:70:70:70:70:70:70:
8a:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:
a0:20:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:
a0:20:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:
20:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:
f0:00:1f:1f:00:1f:1f:00:1f:1f:00:1f:1f:00:1f:1f:00:
c3:f0:a3:1f:1f:00:1f:1f:00:1f:1f:00:1f:1f:00:1f:1f:
70:70:70:70:70:70:70:70:70:70:70:70:70:70:70:70:
a9:00
-----END-----
Format: X.509 (DER)

```

Figura 4: Informação relativa ao respetivo certificado

```

-----BEGIN-----
X509v3 extensions:
X509v3 Key Usage: critical
Certificate Sign, CRL Sign
X509v3 Basic Constraints: critical
CA:TRUE, pathlen:0
X509v3 Certificate Policies:
Policy: 0.2.216.1.100.42.1.200.2
CPS: http://cp.kgyhsz.gov.hu
User Notice:
Explicit Text:

Authority Information Access:
CA Issuers - URI:http://aia.kgyhsz.gov.hu/KGYHSZ_CA_20091210.cer
OCSP - URI:http://ocsp.kgyhsz.gov.hu/ocsp/

X509v3 CRL Distribution Points:

Full Name:
URI:http://crl.kgyhsz.gov.hu/KGYHSZ_CA_20091210.crl

X509v3 Subject Alternative Name:
email:info@hiteles.gov.hu
X509v3 Authority Key Identifier:
keyID:FC:9C:E6:B0:0A:EA:1F:D7:FA:7E:2E:20:05:68:5C:07:4A:C2:E2

X509v3 Subject Key Identifier:
23:50:B8:37:C7:0C:4E:FF:57:81:5C:9B:2C:E6:D8:A6:58:3F:C0:D1
Signature Algorithm: sha256WithRSAEncryption
aa:b6:88:cd:19:8f:01:90:27:24:c8:c2:fd:04:9e:9d:b2:23:
cc:d8:6e:af:6b:5e:b6:4a:a4:0a:e2:d3:49:08:90:dc:fa:4c:
26:fc:29:ae:dd:20:a2:25:98:40:a9:c5:32:18:11:d4:11:8f:
e6:18:ef:cc:8b:10:3d:44:18:0c:cf:70:df:8e:99:0e:e4:e6:
22:2e:92:70:8b:23:70:f9:a7:88:31:59:69:01:d7:38:6b:9f:
63:7d:b1:4d:a4:90:7c:ef:41:02:4d:98:ef:d3:11:bb:2f:c9:
8e:77:ee:d8:7d:68:ec:99:b4:f1:88:8b:0f:d2:7f:0c:e8:0e:
24:f1:f2:c0:07:dd:12:a4:5c:45:28:5c:24:ee:91:fb:3a:9b:
99:13:c3:f9:97:37:1c:b8:96:af:ab:64:a4:a9:91:28:b1:aa:
b8:2c:46:40:84:67:aa:77:a4:b1:31:32:56:f7:58:b5:34:5a:
e7:65:72:cb:87:03:1d:32:58:0d:4a:52:92:f8:3c:fe:05:06:
19:f8:39:ed:c2:d2:a0:a3:82:3d:ff:3c:ad:9c:bd:e0:49:35:
2f:a1:43:1b:94:49:a8:0b:1d:ea:ba:f8:fe:d3:93:64:82:d5:
59:54:2c:84:8c:5d:27:eb:bf:09:1b:1d:18:a5:b9:ba:1a:7a:
eb:2e:7a:91
-----END-----

```

Figura 5: Informação relativa ao respetivo certificado

Posto isto podemos concluir que os algoritmos utilizado são os mais adequados, tanto a curto como a médio prazo uma vez que tanto a encriptação via *RSA* complementado com o *SHA-256* são dos algoritmos criptográficos mais seguros atualmente.