uminho-miei-engseg / 1718-G1

Branch: master ▾ 1718-G1 / Aula 2 / Create new file | Upload files | Find file | History

joaomiguel94 Delete aula2.md — Latest commit da812f0 11 hours ago

..

Readme.md	Update Readme.md	11 hours ago
antes.png	Add files via upload	11 hours ago
chave_usada.png	Add files via upload	11 hours ago
depois.png	Add files via upload	11 hours ago
divide_segredo.png	Add files via upload	11 hours ago
funcao.png	Add files via upload	11 hours ago
gera_cert.png	Add files via upload	11 hours ago
gera_chave.png	Add files via upload	11 hours ago
recover.png	Add files via upload	11 hours ago

Readme.md

1. Números aleatórios/pseudoaleatórios

Resposta P1.1

Ao executar o comando `head -c 1024 /dev/random | openssl enc -base64` , notou-se que o mesmo não mostra qualquer output. Isto deve-se ao facto de este ser bloqueado já que o diretório `/dev/random` é a fonte de entropia. Quando o pool de entropia está vazio, operações de leitura no diretório serão bloqueadas até que seja obtido ruído adicional do ambiente.

O comando `head -c 1024 /dev/urandom | openssl enc -base64` já não é bloquedo, pois o diretório `/dev/urandom` reutiliza o pool interno para produzir mais bits pseudo-aleatórios, o que é menos seguro, contudo é bastante mais rápido que o anterior.

(FONTE: https://pt.wikipedia.org/wiki//dev/random)

```
root@CSI:~# head -c 1024 /dev/random | openssl enc -base64

^C
root@CSI:~# head -c 1024 /dev/urandom | openssl enc -base64
Ng5V7lq1J6SyNXBHQtuFX2rp/gviTP7U0igu7hEbPo6bliR5/7TwC7njiKaRSDSZ
x0HwnSt3DunxllqyR5kv041jQfJ6dsmcN6LR9NrpmwlQxMKmKLeub6wAZMKppXzp
oHsAcU+YlkUgYJIQR6yQ+Ew3eK5S4FTqsCKfKBTtLydN62y4H/6rivN2ry7mbXxb
4rBK1QoJPA5A1WYPmw6uo37gAzOxideVEHS93Dox2vwJyVR4npFDRiZS4JcFiVRF
Zgej6/mm2HRbp4mS2iYLIXV36dZnrN5FbSjk2aT28Tn32yGlke3ryXEx47oPdLke
STMgs5PdVpqm46ZVu4lVmKTCExOoO5hHub+7Pk4dLQ5JoXfxdjlvcGn6PItHyTaX
16dRig8+Rwjz8Dl9yP9r6WofjaqhB07qeWJPn+qE3RR4sZTFZKNuVsrIoZ2wp5Hl
csqnJobbzCy9zVc6Jh6riXB5P5n2LgGyCltXTJOh4qgQSErbZfwhoaxjhp5Z1mBZ
3sfSCM5I1BqFHDYN6cNLpkF8d8qmymcDlI6BgwpDCj32Xtp2O1XqSh5QtDosyA9Q
Hgsb8Byibi9wNH3VvSC2Bh/DmtLpUV+Q4Y5qXfYe+bFnNbf9un+/L7hTx4+QJ4sY
8/ZURcxXf2/2Io9dioKQBTj+S29pQK05k6qEi0fUB+JCmtF8CwcBanHmZeNYbkGC
50oXbivPACqXHUJiyihKz23DcUbb7WZeGQyrKUU1m+YXrPjO9GoRMvY4yXwtX/TN
JJNVKXKJ8iCZjf/Kd8geaNlXxJWhq5UmTKeBcnA7+0aslHNG7MzJ8OuQxNK31WcE
bS/hjNP6kveYPD6ELog+0/43H1KSJRzxfdMJG02kySQmP4bCLXE5eF7HdQiv1iNy
NoqenYCpnUXt8N9bLOOP3n5ufiAuVcYhGicHZLH9KXd6glHILyQJuHZ9ofRqKgrl
gKyF0mAw7368OPUWwm9oZTKdfYuidxM2CCMizKDaRJsErHv5FsZL/2YX14xZHQ8y
QpYLWEIemrhIL+NQVoYnjaU2lasX+GafvIS3agLrgJE31gny4Xq2xxGNHpKE35v+
qEeRGa+7w9NJ6jqnSf1LOABkSeMbLpF4WBYFdSf1Y9IdCxFjijGDToLJIxEPMicE
+PZoNgSzxIxeO786MyAWhA5jatxNSEcPO00IcmeiX3uuRi0gn/pZT5Z2a0xFn4sf
Z1lrtaK6zvRZRU2xmGyo/xuKToxN6Gdm9HOIA6LwxYuo7YdxJYbRr3r6gkhVPYjz
sepLm4ROdQOoQHSTCCptWacID/am5KVSAi150EJGGE3IcIDJKlX1vIusLm0zr9Z3
zxCCymHRuMuzO85I/rpoAQ==
```

Resposta P1.2

Após instalada a package haveged na máquina virtual, o comando `head -c 1024 /dev/random | openssl enc -base64` já dá um output. a package corrige as situações decritas anteriormente

```
root@CSI:~# head -c 1024 /dev/random | openssl enc -base64
WhPU5yWCi1dvWuo6P/5Ght4nbIg2+j4KPw9ia0Kd88tv6hEO5N6+iKqZLphrJDBJ
S+MJIo20YPlvr9FgnOde7My9Duha6kSHjim/z9pxsZd9TqXS2ggMlRL1zLzOT9b8
cZd1osOqy7WroO9fpyBv8x1z38E/sRVir0h7zi97admXMewsBQ3RRWO6dAOjGWom
KxmXlh4NeAy9SjNPmQtSbvyd8/cI4/XKYsbE4lo1S2uW9RDqQUi3QhXTUMzQoINW
ZHexEKaVLUN52nNxnG30NHKD/vTQmHfmquM5yrsoB4pLZhyuIdZqB4ApLaKy5OJK
9QRA25r+JSPkM5Hu/OZwCfJWacWaGr/5OK2/vCW9So8295er23vRGkkPz9p44xLv
4/I7LflYaFmpXQV46r4dXuJMxAHR9ZW7PXD11y59sV8/f9zMg8jXE1UGotnMXyFH
U+NM3M9grMS/j9xrg4PdWivqn6g8xxgHMj1kXbk1ru12/sw7oES8L1jlgZNR8s2L
86qhxU7yCxHIk+8IZs36IHCRZJjbIylNU8pnUsz09uBEG6Fdwt1m+gjAadfiEVev
Ipju89K5Px8wjcpby3e9ThE/GPJA6XxEjh3HhhFf2IRb18LJYPzW7UrE2/gH1D53
Eld8B2wuKwLy+Skt+Za+j/bzfSlwl4FlULfnIaY94AKh/oP2gymWuPfwTw+zlDNx
nLLtO9lVTjW2RwhTSGTyto+WYCmKIq77DxymdAAX+YKaZ3vJxuNUZuNsZfujjqZG
xQZ4dY1ctY9PzC8t9kf/dqwTQPt7FZfko2+tA6cJunWRugUa7lrI9TqE2vh01L4T
GHJszMHvXHm06Iu99Tali+Crz7pM2t8mDlQjsQX1cDC46z8iWyxWM/ycw+/LUF+J
QnvPMY7EOlGKZ0vilwW5q8qmaUEr7hHeZDMdLlxsAjOJE+bLekhMAF0hasb4+tjt
ycKqBg3/jseSo27MJ+cTmkApOllD/ZJ7DkM68txe4HRSeMad7OMUIaH7KfnERDal
NzhpDpYPIemEelSWc1k9s3cl3/69QlXSjvlt2jyszGl4LKaTXglZt8e350OOdmNQ
egBMfjSX56r6tvz0BdaiQp+zj6FvaY7x6Nje+cKLJMwdYf7aZbhrEn64S7EOGVjr
/zeifM9yDT/0sdGu9ZovR41Bom55RF40M6lsyTETLxucQo5Du/vK5AXPpO8raeW3
K2reZFQhDhIX+MvldyweE5JwDfwW+uMbE9X41+0Zme+ugiST+9TN0LMnQ2yZNm8n
IQSKDGs6ukIa+WHUd2nOtS5ZfIL2LqULwvwY4Xr6x49PJ68SpylSKl+9pmDifbS8
+BRXnrvJo7g2+TcQyuydTQ==
root@CSI:~# head -c 1024 /dev/urandom | openssl enc -base64
JKK1ygLHuZ+vb8zpSkCZAM1LC6uefgorstlEGrVZJMmNJ9JcKsn2bSvnUEQPf6yO
JZpIfpysfpvrkyns2PaqbzJr77zqtwAQWSkTeGSkxutkYa8sKmfzCeQTSJs89ddU
O/kPA+mT9np0Y07tWSpr5povaN4cVpXmUafBa9+G9xZ6wOUcAPFqm2RSq7+QeTwH
XfNU6YMCMi7Kg/JR5Dpa0WcksaGfpFaLnKscybLDtQaJDOoqVR4Uu2Mvibtdq/fC
Z3jpYIhzOuXWbPaLF6c1YsJIh1S0yHJGfziuLUwwfosZbjel5EnG6NO+0E/3N+IR
csoOuINrAQ/8D2OqNGOR9tpogLtAd0tCdWQ4Zx3l83JVFtVDdv+oZyCgtUzCK3vb
dTPSOGM6oxV0gWJW4Z7fM4IBkaheweCEcL8KMNdh/N3/l7Ol+BusOj2O7E+kGsoQ
jTOmoVmL6OaY8PhUFpsJKO8ubio1hs/JPCYn5kQwPrdPNziGl6rpojCRlOhpC9OD
tAgqaHgPmyw+Q6OoluWGjo3WhDOHw0lzC2wT8Kwi3QuMIqX0ZsEU8oqBeD8ZUJqs
X738NLEmArpeRpGGCdSO9G9bZyKuWukJ5yx/X1BXcXUqWTlLM4F1W9UZ+FRaDC2l
qWSJHZ8nTAbNKSwWZNXls7T5IwmexbwaX5fe/2yppvpVL76cHNgYuiT8bsOSwBcX
Xhy7XtArmgiXHcXJErwDM2cJeqSm7+Eg5N1nIWYNO9NEgbw7RSSpg/MJ5RtYFZeT
DPDN/TjKaq1cKTbDj11RQ+hEpO6PTxAGdvVrYuOlrRImFwdsgr9r+ex5HSWhSL8V
5WBNTwKhyi0LiPKeJ9uPEcWcKE6BAOIMLc+9utCX4a4zdOAIxeacZGrOLE73HlGk
cqi5JorBKHKeSN6m9NPMeYi0csgsEjEJjg6uLmgaG25XPp2DVTUPASw5FbUs1/xt
cx5xlLsnoQmZBhemm0WbxXaaaIaycsXQEAEq7daLCOdy5tLmObwRPhLIm0cU4Y55
wj0YHb4zix3TJkdejTZ+SfvBVP0EW+z99sa5KG4UbU5uN0pK9wm8zghttPy69gS6
jyyHyHtzUGRyUs3VWSxcqNkLQ6ET3u2UngyIobufPaIdWz7euhjMfHwZNYAx6J/u
lAugw6A9kW20rcM7R2OCllcOOHc3k6QxCcc2po04M2gJLjRijqSYB1LRGDEG1nAu
tnByshCp1j2Y5krVavAkfwt5fHbNkiGXpjBflym1kBMTv5d38Vgm5yJNtzN3ocet
xETl5b4LzMglU23cwNR2hLfbQD4tTNt8/wmOFQDyaJNQANS2f31ApZthFlw9Ihxq
mr24XMzvOxjVBBsXF1UF1Q==
root@CSI:~#
```

Resposta P1.3

É na função `generateSecret` no ficheiro `shamirsecret.py` que é especificado como são gerados os segredos (letras e digitos ascii), como mostra a figura abaixo

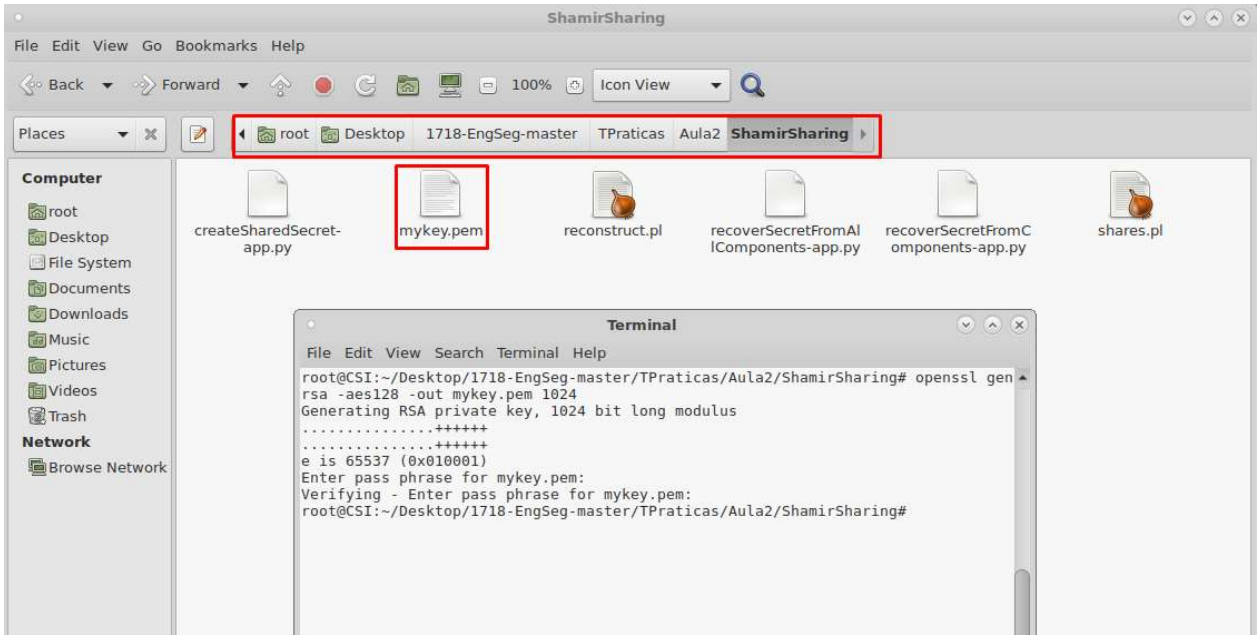
```python
#Cripto-4.4.0
def generateSecret(secretLength):
    """
    This function generates a random string with secretLength characters (ascii_letters and digits)
    Args:
        secretLength (int): number of characters of the string
    Returns:
        Random string with secretLength characters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

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

Resposta P2.1

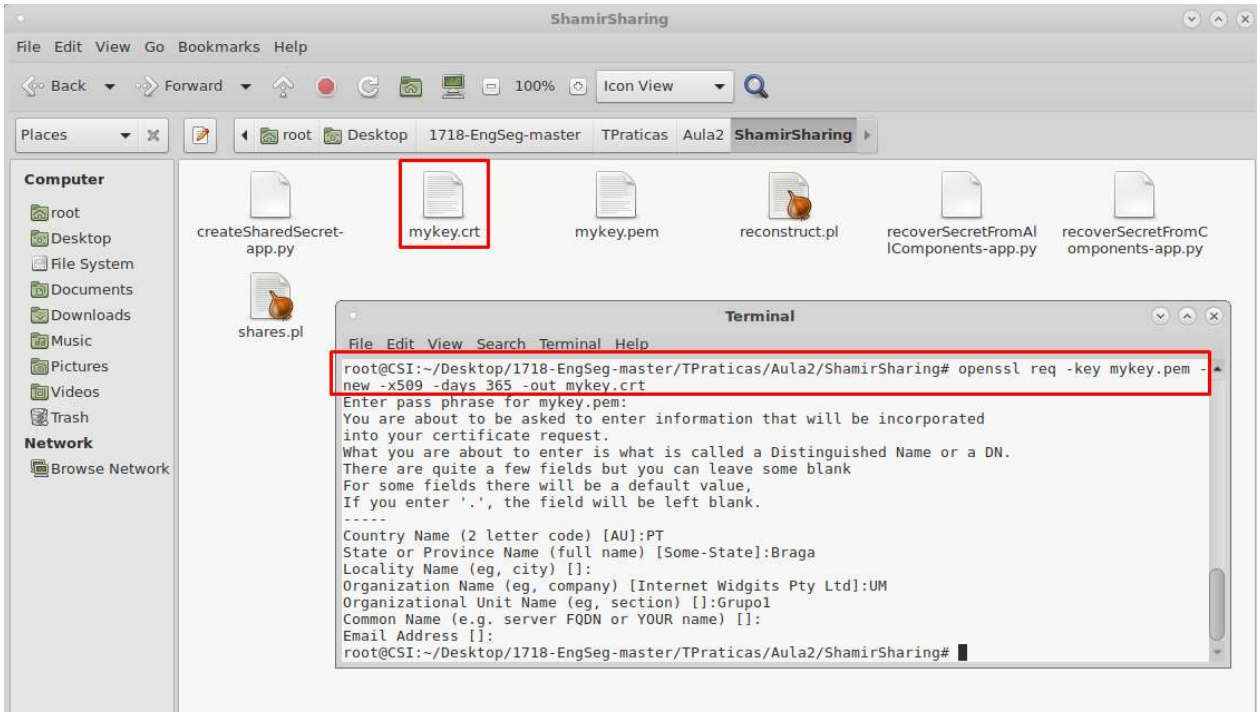
A

A chave privada faz parte dos argumentos do programa, por isso deve-se gerar uma nova antes da execução, através do comando `openssl genrsa -aes128 -out mykey.pem 1024` .



Depois executa-se o comando para a chave criada (`python createSharedSecret-app.py 7 3 1 mykey.pem`)

Para verificar se as 3 partes conseguem recuperar o segredo original, deve-se primeiro criar um certificado para a chave `mykey.pem` . Para isso usa-se o comando `openssl req -key mykey.pem -new -x509 -days 365 -out mykey.crt` .



No final executa-se `python recoverSecretFromComponents-app.py 3 1 mykey.crt` .

S

B

A diferença está nas partes que são necessárias para reconstruir o segredo. No `recoverSecretFromAllComponents` o segredo é reconstruído se todos os componentes forem inseridos corretamente. Já no `recoverSecretFromComponents` apenas são necessários os componentes especificados (`quorum`).

Isto pode ser aplicado numa universidade, por exemplo.

- Se for necessário validar apenas alguns alunos de um perfil, usa-se o `recoverSecretFromComponents`
- Para todos os alunos do perfil, usa-se o `recoverSecretFromAllComponents`

3. Authenticated Encryption

Enc

```
secret = enc(message, key);                    // key =  dia.mes.ano
authentic = hmac(HMAC_SHA_256_key, secret);
cyphertext = send(secret:authentic);
```

Dec

```
recieve(cyphertext);
authentic, secret = separate(cyphertext) HMAC_256
if (hmac(HMAC_SHA_256, secret) == authentic)
        dec(secret, key);
else
        exit(1);
```

4. Algoritmos e tamanhos de chaves

Austria

A-Trust Gesellschaft für Sicherheitssysteme im elektronischen Datenverkehr GmbH

a-sign-Premium-Sig-01 (key no. 1)

(FONTE: https://webgate.ec.europa.eu/tl-browser/#/tl/AT/0/4)

1. Chave utilizada

Base 64-encoded

CN=a-sign-Premium-Sig-01, OU=a-sign-Premium-Sig-01, O=A-Trust Ges. f. Sicherheitssysteme im elektr. Datenverkehr GmbH, C=AT

MIIEvTCCA6WgAwIBAgICGBQwDQYJKoZIhvcNAQEFBQAwgc8xCzAJBgNVBAYTAkFUMYGLMIGIBgNVBAoegYAAQQAtAFQAcgB1AHMAdAAgAEcAZQBzAC4AIABmAPwAcgAgAFMAaQBjAGgAZQByAGgAZQBpAHQAcwBzAHkAcwB0AGUAbQBlACAAaQBtACAAZQBsAGUAawB0AHIALgAgAEQAYQB0AGUAbgB2AGUAcgBrAGUAaAByACAARwBtAGIASDEYMBYGA1UECxMPQS1UcnVzdC1RdWFsLTAxMRgwFgYDVQQDEw9BLVRydXN0LVF1YWwtMDEwHhcNMDMwMTIyMjMwMDAwWhcNMDYwMTIyMjMwMDAwWjCBlzELMAkGA1UEBhMCQVQxSDBGBgNVBAoTP0EtVHJ1c3QgR2VzLiBmLiBTaWNoZXJoZWl0c3N5c3RlbWUgaW0gZWxla3RyLiBEYXRlbnZlcmtlaHIgR21iSDEeMBwGA1UECxMVYS1zaWduLVByZW1pdW0tU2InLTAxMR4wHAYDVQQDExVhLXNpZ24tUHJlbWl1bS1TaWctMDEwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDAlOjUod+36uDr17MyREqpilqx41bJsrt+laqXBVuFi9G8p1efHY4XVkQ43aob0Pegt/+ZOWeCxPKnKwH3+D8nZMP3MyIqoiDbPYL7u0BTilP7/gtrinopPR4mNEvxuMb1bkPpeghH26eT/+14XvEgxZRS+fipheJGjCWRlOEWjZ2nu4aNWtQqz9nQToV76BMwuU4xqB2+xFdOgSbTNIvkxs0mKAzNwTb7OwYfjXp8c13+ppkUB4wWzsPkW1H1IBrpPUjGFEt+IdSIbyvWdaFb/WAymArcIOPXos1erief5RDSXacw6AgJFxUzZHARqB3Wqki0uKznOqOOI4UZszI9AgMBAAGjgdgwgdUwDwYDVR0TAQH/BAUwAwEB/zARBgNVHQ4ECgQIQNfznhuHOswwEwYDVR0jBAwwCoAISzyMHYXpb60wJAYIKwYBBQUHAQMEGDAWMAgGBgQAjkYBATAKBggrBgEFBQcLATAOBgNVHQ8BAf8EBAMCAQYwZAYDVR0fBF0wWzBZoFegVYZTbGRhcDovL2xkYXAuYS10cnVzdC5hdC9vdT1BLVRydXN0LVF1YWwtMDEsbz1BLVRydXN0LGM9QVQ/Y2VydGlmaWNhdGVyZXZvY2F0aW9ubGlzdD8wDQYJKoZIhvcNAQEFBQADggEBACflLIJ51oBizo44aVKiC2BTL3uraHwNzmx0Bl82NtSoC+l6K6jU5wir2oGMq50jb/645RHH6CiPcn9Z8/KzO+EAnwLSjsaupFMBRSTnyQesPoIRQ9UPFWecVZjjsfNmuydvNGr2aRJnaaoKR+8SLYdIEqrAG44IWBCvdHZuNvMxH7qh97Enr7oE6Ur6XuKlpRkRE6F1jGUEW1OzHbzYpp78N/HV2FEf37SsIIRFycRwog/LGAAHVKr16BzfVJwa/uaGSLm3reXqe/6eJnI8PmXHqLkHPW5UEpBhAldBHq2o0Mnrn3jrAE06h6kH34Zxrv7xBTnUjZsvIL9/bwGkMWY=

2. Algoritmo usado

```
SEQUENCE {
   SEQUENCE {
      [0] {
         INTEGER 0x02 (2 decimal)
      }
      INTEGER 0x2115 (8469 decimal)
      SEQUENCE {
         OBJECTIDENTIFIER 1.2.840.113549.1.1.5 (sha1WithRSAEncryption)
         NULL
      }
      SEQUENCE {
         SET {
            SEQUENCE {
               OBJECTIDENTIFIER 2.5.4.6 (countryName)
               PrintableString 'AT'
            }
         }
         SET {
            SEQUENCE {
               OBJECTIDENTIFIER 2.5.4.10 (organizationName)
               [1e]
0041002d00540072007500730074002000470065007300 2e0020006600fc00720020005300690063006800650072006800650069007400730073
            }
         }
         SET {
            SEQUENCE {
               OBJECTIDENTIFIER 2.5.4.11 (organizationalUnitName)
               PrintableString 'A-Trust-Qual-01'
            }
         }
         SET {
            SEQUENCE {
               OBJECTIDENTIFIER 2.5.4.3 (commonName)
               PrintableString 'A-Trust-Qual-01'
            }
         }
      }
      SEQUENCE {
         UTCTime '030122230000Z'
         UTCTime '060122230000Z'
      }
      SEQUENCE {
         SET {
            SEQUENCE {
               OBJECTIDENTIFIER 2.5.4.6 (countryName)
               PrintableString 'AT'
            }
         }
         SET {
```

```
            SEQUENCE {
               OBJECTIDENTIFIER 2.5.4.10 (organizationName)
               PrintableString 'A-Trust Ges. f. Sicherheitssysteme im elektr. Datenverkehr GmbH'
            }
         }
         SET {
            SEQUENCE {
               OBJECTIDENTIFIER 2.5.4.11 (organizationalUnitName)
               PrintableString 'a-sign-Premium-Sig-01'
            }
         }
         SET {
            SEQUENCE {
               OBJECTIDENTIFIER 2.5.4.3 (commonName)
               PrintableString 'a-sign-Premium-Sig-01'
            }
         }
      }
      SEQUENCE {
         SEQUENCE {
            OBJECTIDENTIFIER 1.2.840.113549.1.1.1 (rsaEncryption)
            NULL
         }
         BITSTRING
0x3082010a0282010100c1fccf6ae7aec57a0f31b476d884c860f0b75b49f7a138121dae44917440bc7b6d6d7c5bf81c29b3c681194682c792afc
 : 0 unused bit(s)
      }
      [3] {
         SEQUENCE {
            SEQUENCE {
               OBJECTIDENTIFIER 2.5.29.19 (basicConstraints)
               BOOLEAN TRUE
               OCTETSTRING 30030101ff
            }
            SEQUENCE {
               OBJECTIDENTIFIER 2.5.29.14 (subjectKeyIdentifier)
               OCTETSTRING 04084379d3f07f719ab2
            }
            SEQUENCE {
               OBJECTIDENTIFIER 2.5.29.35 (authorityKeyIdentifier)
               OCTETSTRING 300a800841d8c9066599388c
            }
            SEQUENCE {
               OBJECTIDENTIFIER 1.3.6.1.5.5.7.1.3
               OCTETSTRING 30163008060604008e460101300a06082b06010505070b01
            }
            SEQUENCE {
               OBJECTIDENTIFIER 2.5.29.15 (keyUsage)
               BOOLEAN TRUE
               OCTETSTRING 03020106
            }
            SEQUENCE {
               OBJECTIDENTIFIER 2.5.29.31 (cRLDistributionPoints)
               OCTETSTRING
3081b63059a057a05586536c6461703a2f2f6c6461702e612d74727573742e61742f6f753d412d54727573742d5175616c2d30312c6f3d412d547

            }
         }
      }
   }
   SEQUENCE {
      OBJECTIDENTIFIER 1.2.840.113549.1.1.5 (sha1WithRSAEncryption)
      NULL
   }
   BITSTRING
0x582c068f6967e1680c22e78f055b167cb2164ea875c4bc56198cf6e68678a606ab44d436c4457648b0baaa4ad09b408f49bddffa90346cfb345
 : 0 unused bit(s)
}
```