Asymmetric Crypto with OpenSSL

Giuseppe Persiano

Università di Salerno

November, 2020

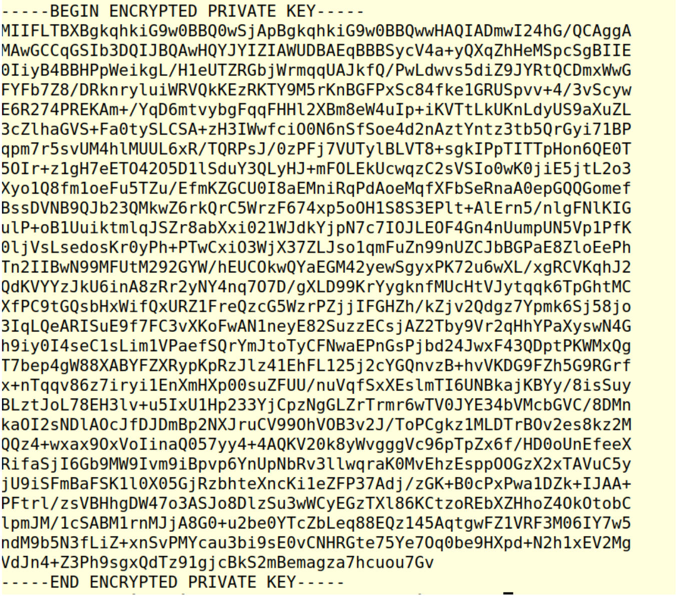
RSA with OpenSSL

Key Generation

- `openssl genpkey`: generate private key
- `-aes256`: encryption algorithm used to armor the private key
- `-algorithm RSA`: public-key encryption scheme
- `-pkeyopt rsa_keygen_bits:2048` specify option for the key. In this case the number of bits
- `-out:` file that will store the key

The key is stored in the `PKCS#8` format
Encrypted using `PBKDF2` with `hmacWithSHA256` as a PRF

Public exponent: `65537 (0x10001)`
`-pkeyopt rsa_keygen_pubexp:` to modify



```
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIIFLTBXBgkqhkiG9w0BBQ0wSjApBgkqhkiG9w0BBQwwHAQIADmwI24hG/QCAggA
MAwGCCqGSIb3DQIJBQAwHQYJYIZIAWUDBAEqBBBSycV4a+yQXqZhHeMSpcSgBIIE
0IiyB4BBHPpWeikgL/H1eUTZRGbjWrmqqUAJkfQ/PwLdwvs5diZ9JYRtQCDmxWwG
FYFb7Z8/DRknryluiWRVQkKEzRKTY9M5rKnBGFPxSc84fke1GRUSpvv+4/3vScyw
E6R274PREKAm+/YqD6mtvybgFqqFHHl2XBm8eW4uIp+iKVTtLkUKnLdyUS9aXuZL
3cZlhaGVS+Fa0tySLCSA+zH3IWwfciO0N6nSfSoe4d2nAztYntz3tb5QrGyi71BP
qpm7r5svUM4hlMUUL6xR/TQRPsJ/0zPFj7VUTylBLVT8+sgkIPpTITTpHon6QE0T
5OIr+z1gH7eETO42O5D1lSduY3QLyHJ+mFOLEkUcwqzC2sVSIo0wK0jiE5jtL2o3
Xyo1Q8fm1oeFu5TZu/EfmKZGCU0I8aEMniRqPdAoeMqfXFbSeRnaA0epGQQGomef
BssDVNB9QJb23QMkwZ6rkQrC5WrzF674xp5oOH1S8S3EPlt+AlErn5/nlgFNlKIG
ulP+oB1UuiktmlqJSZr8abXxi021WJdkYjpN7c7IOJLEOF4Gn4nUumpUN5Vp1PfK
0ljVsLsedosKr0yPh+PTwCxiO3WjX37ZLJso1qmFuZn99nUZCJbBGPaE8ZloEePh
Tn2IIBwN99MFUtM292GYW/hEUCOkwQYaEGM42yewSgyxPK72u6wXL/xgRCVKqhJ2
QdKVYYzJkU6inA8zRr2yNY4nq7O7D/gXLD99KrYygknfMUcHtVJytqqk6TpGhtMC
XfPC9tGQsbHxWifQxURZ1FreQzcG5WzrPZjjIFGHZh/kZjv2Qdgz7Ypmk6Sj58jo
3IqLQeARISuE9f7FC3vXKoFwAN1neyE82SuzzECsjAZ2Tby9Vr2qHhYPaXyswN4G
h9iy0I4seC1sLim1VPaefSQrYmJtoTyCFNwaEPnGsPjbd24JwxF43QDptPKWMxQg
T7bep4gW88XABYFZXRypKpRzJlz41EhFL125j2cYGQnvzB+hvVKDG9FZh5G9RGrf
x+nTqqv86z7iryi1EnXmHXp00suZFUU/nuVqfSxXEslmTI6UNBkajKBYy/8isSuy
BLztJoL78EH3lv+u5IxU1Hp233YjCpzNgGLZrTrmr6wTV0JYE34bVMcbGVC/8DMn
kaOI2sNDlAOcJfDJDmBp2NXJruCV99OhVOB3v2J/ToPCgkz1MLDTrBOv2es8kz2M
QQz4+wxax9OxVoIinaQ057yy4+4AQKV20k8yWvgggVc96pTpZx6f/HD0oUnEfeeX
RifaSjI6Gb9MW9Ivm9iBpvp6YnUpNbRv3llwqraK0MvEhzEsppOOGzX2xTAVuC5y
jU9iSFmBaFSK1l0X05GjRzbhteXncKi1eZFP37Adj/zGK+B0cPxPwa1DZk+IJAA+
PFtrl/zsVBHhgDW47o3ASJo8DlzSu3wWCyEGzTXl86KCtzoREbXZHhoZ4OkOtobC
lpmJM/1cSABM1rnMJjA8G0+u2be0YTcZbLeq88EQz145AqtgwFZ1VRF3M06IY7w5
ndM9b5N3fLiZ+xnSvPMYcau3bi9sE0vCNHRGte75Ye7Oq0be9HXpd+N2h1xEV2Mg
VdJn4+Z3Ph9sgxQdTz91gjcBkS2mBemagza7hcuou7Gv
-----END ENCRYPTED PRIVATE KEY-----
```

Not very informative

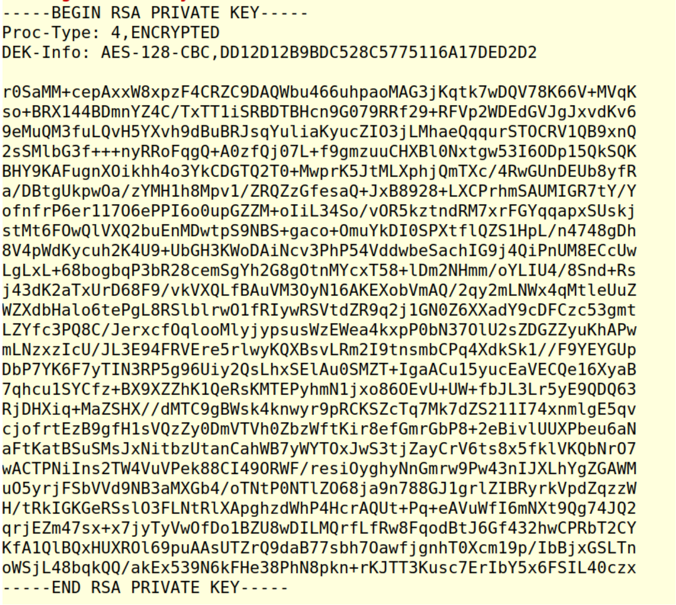
Key Generation: Obsolete

- `openssl genrsa`
- `-aes128`: encryption algorithm used to armor the private key
- `-out` file that will store the key
- `len` in bits of the key

```
openssl genrsa -aes128 -out giuperPrivateRSA.pem 2048
```

The key is stored in the `PKCS#1` format

Public exponent: `65537 (0x10001)`



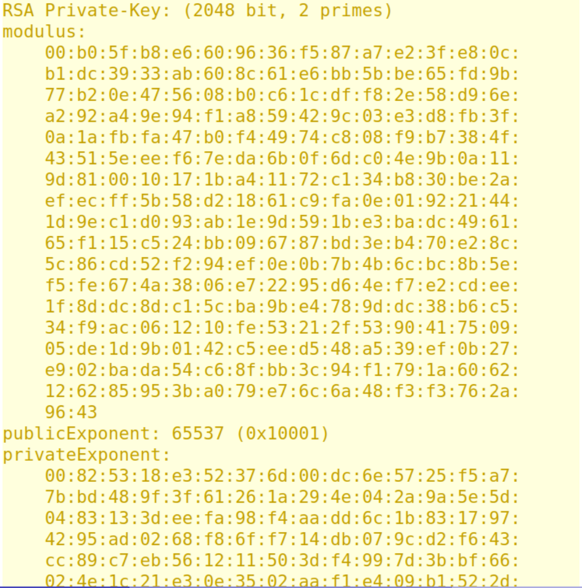
```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,DD12D12B9BDC528C5775116A17DED2D2

r0SaMM+cepAxxW8xpzF4CRZC9DAQWbu466uhpaoMAG3jKqtk7wDQV78K66V+MVqK
so+BRX144BDmnYZ4C/TxTT1iSRBDTBHcn9G079RRf29+RFVp2WDEdGVJgJxvdKv6
9eMuQM3fuLQvH5YXvh9dBuBRJsqYuliaKyucZIO3jLMhaeQqqurSTOCRV1QB9xnQ
2sSMlbG3f+++nyRRoFqgQ+A0zfQj07L+f9gmzuuCHXBl0Nxtgw53I6ODp15QkSQK
BHY9KAFugnXOikhh4o3YkCDGTQ2T0+MwprK5JtMLXphjQmTXc/4RwGUnDEUb8yfR
a/DBtgUkpwOa/zYMH1h8Mpv1/ZRQZzGfesaQ+JxB8928+LXCPrhmSAUMIGR7tY/Y
ofnfrP6er117O6ePPI6o0upGZZM+oIiL34So/vOR5kztndRM7xrFGYqqapxSUskj
stMt6FOwQlVXQ2buEnMDwtpS9NBS+gaco+OmuYkDI0SPXtflQZS1HpL/n4748gDh
8V4pWdKycuh2K4U9+UbGH3KWoDAiNcv3PhP54VddwbeSachIG9j4QiPnUM8ECcUw
LgLxL+68bogbqP3bR28cemSgYh2G8gOtnMYcxT58+lDm2NHmm/oYLIU4/8Snd+Rs
j43dK2aTxUrD68F9/vkVXQLfBAuVM3OyN16AKEXobVmAQ/2qy2mLNWx4qMtleUuZ
WZXdbHalo6tePgL8RSlblrwO1fRIywRSVtdZR9q2j1GN0Z6XXadY9cDFCzc53gmt
LZYfc3PQ8C/JerxcfOqlooMlyjypsusWzEWea4kxpP0bN37OlU2sZDGZZyuKhAPw
mLNzxzIcU/JL3E94FRVEre5rlwyKQXBsvLRm2I9tnsmbCPq4XdkSk1//F9YEYGUp
DbP7YK6F7yTIN3RP5g96Uiy2QsLhxSElAu0SMZT+IgaACu15yucEaVECQe16XyaB
7qhcu1SYCfz+BX9XZZhK1QeRsKMTEPyhmN1jxo86OEvU+UW+fbJL3Lr5yE9QDQ63
RjDHXiq+MaZSHX//dMTC9gBWsk4knwyr9pRCKSZcTq7Mk7dZS211I74xnmlgE5qv
cjofrtEzB9gfH1sVQzZy0DmVTVh0ZbzWftKir8efGmrGbP8+2eBivlUUXPbeu6aN
aFtKatBSuSMsJxNitbzUtanCahWB7yWYTOxJwS3tjZayCrV6ts8x5fklVKQbNrO7
wACTPNiIns2TW4VuVPek88CI49ORWF/resiOyghyNnGmrw9Pw43nIJXLhYgZGAWM
uO5yrjFSbVVd9NB3aMXGb4/oTNtP0NTlZO68ja9n788GJ1grlZIBRyrkVpdZqzzW
H/tRkIGKGeRSslO3FLNtRlXApghzdWhP4HcrAQUt+Pq+eAVuWfI6mNXt9Qg74JQ2
qrjEZm47sx+x7jyTyVwOfDo1BZU8wDILMQrfLfRw8FqodBtJ6Gf432hwCPRbT2CY
KfA1QlBQxHUXROl69puAAsUTZrQ9daB77sbh7OawfjgnhT0Xcm19p/IbBjxGSLTn
oWSjL48bqkQQ/akEx539N6kFHe38PhN8pkn+rKJTT3Kusc7ErIbY5x6FSIL40czx
-----END RSA PRIVATE KEY-----
```

Not very informative

What is in a key

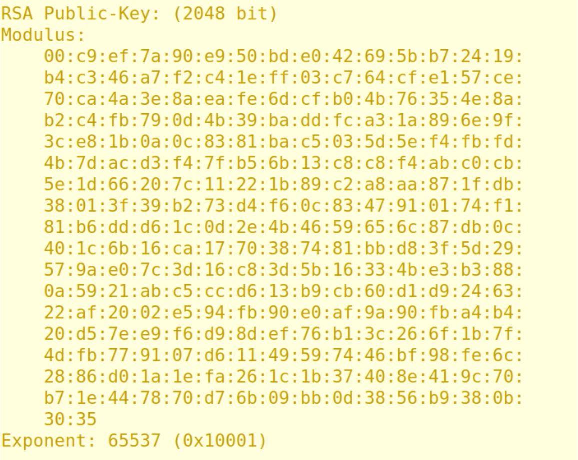
```
openssl rsa -text -in giuperPrivateRSA.pem -noout
```



```
RSA Private-Key: (2048 bit, 2 primes)
modulus:
    00:b0:5f:b8:e6:60:96:36:f5:87:a7:e2:3f:e8:0c:
    b1:dc:39:33:ab:60:8c:61:e6:bb:5b:be:65:fd:9b:
    77:b2:0e:47:56:08:b0:c6:1c:df:f8:2e:58:d9:6e:
    a2:92:a4:9e:94:f1:a8:59:42:9c:03:e3:d8:fb:3f:
    0a:1a:fb:fa:47:b0:f4:49:74:c8:08:f9:b7:38:4f:
    43:51:5e:ee:f6:7e:da:6b:0f:6d:c0:4e:9b:0a:11:
    9d:81:00:10:17:1b:a4:11:72:c1:34:b8:30:be:2a:
    ef:ec:ff:5b:58:d2:18:61:c9:fa:0e:01:92:21:44:
    1d:9e:c1:d0:93:ab:1e:9d:59:1b:e3:ba:dc:49:61:
    65:f1:15:c5:24:bb:09:67:87:bd:3e:b4:70:e2:8c:
    5c:86:cd:52:f2:94:ef:0e:0b:7b:4b:6c:bc:8b:5e:
    f5:fe:67:4a:38:06:e7:22:95:d6:4e:f7:e2:cd:ee:
    1f:8d:dc:8d:c1:5c:ba:9b:e4:78:9d:dc:38:b6:c5:
    34:f9:ac:06:12:10:fe:53:21:2f:53:90:41:75:09:
    05:de:1d:9b:01:42:c5:ee:d5:48:a5:39:ef:0b:27:
    e9:02:ba:da:54:c6:8f:bb:3c:94:f1:79:1a:60:62:
    12:62:85:95:3b:a0:79:e7:6c:6a:48:f3:f3:76:2a:
    96:43
publicExponent: 65537 (0x10001)
privateExponent:
    00:82:53:18:e3:52:37:6d:00:dc:6e:57:25:f5:a7:
    7b:bd:48:9f:3f:61:26:1a:29:4e:04:2a:9a:5e:5d:
    04:83:13:3d:ee:fa:98:f4:aa:dd:6c:1b:83:17:97:
    42:95:ad:02:68:f8:6f:f7:14:db:07:9c:d2:f6:43:
    cc:89:c7:eb:56:12:11:50:3d:f4:99:7d:3b:bf:66:
    02:4e:1c:21:e3:0e:35:02:aa:f1:e4:09:b1:52:2d:
```

Splitting Public and Private Key

- `openssl rsa -pubout -in giuperPrivateRSA.pem -out giuperPublicRSA.pem`
- `openssl rsa -pubin -in giuperPublicRSA.pem -text -noout`



```
RSA Public-Key: (2048 bit)
Modulus:
    00:c9:ef:7a:90:e9:50:bd:e0:42:69:5b:b7:24:19:
    b4:c3:46:a7:f2:c4:1e:ff:03:c7:64:cf:e1:57:ce:
    70:ca:4a:3e:8a:ea:fe:6d:cf:b0:4b:76:35:4e:8a:
    b2:c4:fb:79:0d:4b:39:ba:dd:fc:a3:1a:89:6e:9f:
    3c:e8:1b:0a:0c:83:81:ba:c5:03:5d:5e:f4:fb:fd:
    4b:7d:ac:d3:f4:7f:b5:6b:13:c8:c8:f4:ab:c0:cb:
    5e:1d:66:20:7c:11:22:1b:89:c2:a8:aa:87:1f:db:
    38:01:3f:39:b2:73:d4:f6:0c:83:47:91:01:74:f1:
    81:b6:dd:d6:1c:0d:2e:4b:46:59:65:6c:87:db:0c:
    40:1c:6b:16:ca:17:70:38:74:81:bb:d8:3f:5d:29:
    57:9a:e0:7c:3d:16:c8:3d:5b:16:33:4b:e3:b3:88:
    0a:59:21:ab:c5:cc:d6:13:b9:cb:60:d1:d9:24:63:
    22:af:20:02:e5:94:fb:90:e0:af:9a:90:fb:a4:b4:
    20:d5:7e:e9:f6:d9:8d:ef:76:b1:3c:26:6f:1b:7f:
    4d:fb:77:91:07:d6:11:49:59:74:46:bf:98:fe:6c:
    28:86:d0:1a:1e:fa:26:1c:1b:37:40:8e:41:9c:70:
    b7:1e:44:78:70:d7:6b:09:bb:0d:38:56:b9:38:0b:
    30:35
Exponent: 65537 (0x10001)
```

Removing the armor

```
openssl rsa -in giuperPrivateRSA.pem
          -out giuperPrivateRSANonArmor.pem
```

Encrypting using the RSA public key

Generate a 16-byte AES key and store it in `aaa.txt`

Encrypt it using RSA

```
openssl rsautl -encrypt -oaep -pubin
    -inkey giuperPublicRSA.pem -in aaa.txt -out aaa.txt.cpt
```

Decrypting using the RSA private key

Decrypt the file aaa.txt.cpt

```
openssl rsautl -decrypt -oaep -inkey giuperPrivateRSA.pem
       -in aaa.txt.cpt -out aaa.txt.new
```

Signing using the RSA private key (Obsolete)

Signing the file aaa.txt

```
openssl rsautl -sign -inkey giuperPrivateRSA.pem
       -in aaa.txt -out aaa.txt.sig
```

The output file aaa.txt.sig contains the file aaa.txt and the signature.

Hashing algorithm: MD5

Verifying a signature using the RSA public key (Obsolete)

Verifying the signature `aaa.txt.sig`

```
openssl rsautl -verify -pubin -inkey giuperPublicRSA.pem
       -in aaa.txt.sig -out aaa.txt.vrf
```

The output file `aaa.txt.sig` is assumed to contain the file `aaa.txt` and the signature.

Signing using the RSA private key

Signing the file aaa.txt

```
openssl dgst -sha256 -sign giuperPrivateRSA.pem
       -out aaa.txt.sig aaa.txt
```

The output file aaa.txt.sig does **not** contain the file aaa.txt but only the signature.

Hashing algorithm: to be specified (SHA256 in the example)

Verifying a signature using the RSA public key

Verifying the signature aaa.txt.sig

```
openssl dgst -sha256 -verify giuperPublicRSA.pem
        -signature aaa.txt.sig aaa.txt
```

The output file aaa.txt.sig is not assumed to contain the file aaa.txt but only the signature.

The file aaa.txt containing the document must be specified in the command.

Generating Keys for DSA

A two-step process

- First we generate the parameters

  `openssl genpkey -genparam -algorithm DSA`

  options:

  - length of p in bits: `-pkeyopt dsa_paramgen_bits:  2048`
  - length of 1 in bits: `-pkeyopt dsa_paramgen_q_bits:  256`
  - the hashing algorithm: `-pkeyopt dsa_paramgen_md:  sha256`
  - the file containg the parameters: `-out`

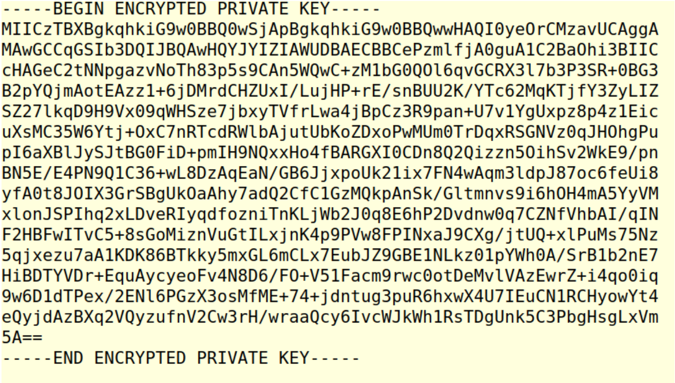
- Then each user generates his/her pair of keys

  `openssl genpkey`

  options:

  - the armor algorithm: `-aes128`
  - the file with the parameters `-paramfile`
  - the file that will contain the private key `-out`

What is in a private key?



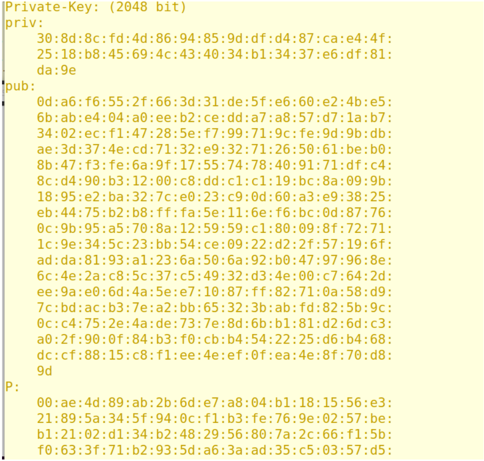
```
-----BEGIN ENCRYPTED PRIVATE KEY-----
MIICzTBXBgkqhkiG9w0BBQ0wSjApBgkqhkiG9w0BBQwwHAQI0yeOrCMzavUCAggA
MAwGCCqGSIb3DQIJBQAwHQYJYIZIAWUDBAECBBCePzmlfjA0guA1C2BaOhi3BIIC
cHAGeC2tNNpgazvNoTh83p5s9CAn5WQwC+zM1bG0QOl6qvGCRX3l7b3P3SR+0BG3
B2pYQjmAotEAzz1+6jDMrdCHZUxI/LujHP+rE/snBUU2K/YTc62MqKTjfY3ZyLIZ
SZ27lkqD9H9Vx09qWHSze7jbxyTVfrLwa4jBpCz3R9pan+U7v1YgUxpz8p4z1Eic
uXsMC35W6Ytj+OxC7nRTcdRWlbAjutUbKoZDxoPwMUm0TrDqxRSGNVz0qJHOhgPu
pI6aXBlJySJtBG0FiD+pmIH9NQxxHo4fBARGXI0CDn8Q2Qizzn5OihSv2WkE9/pn
BN5E/E4PN9Q1C36+wL8DzAqEaN/GB6JjxpoUk21ix7FN4wAqm3ldpJ87oc6feUi8
yfA0t8JOIX3GrSBgUkOaAhy7adQ2CfC1GzMQkpAnSk/Gltmnvs9i6hOH4mA5YyVM
xlonJSPIhq2xLDveRIyqdfozniTnKLjWb2J0q8E6hP2Dvdnw0q7CZNfVhbAI/qIN
F2HBFwITvC5+8sGoMiznVuGtILxjnK4p9PVw8FPINxaJ9CXg/jtUQ+xlPuMs75Nz
5qjxezu7aA1KDK86BTkky5mxGL6mCLx7EubJZ9GBE1NLkz01pYWh0A/SrB1b2nE7
HiBDTYVDr+EquAycyeoFv4N8D6/FO+V51Facm9rwc0otDeMvlVAzEwrZ+i4qo0iq
9w6D1dTPex/2ENl6PGzX3osMfME+74+jdntug3puR6hxwX4U7IEuCN1RCHyowYt4
eQyjdAzBXq2VQyzufnV2Cw3rH/wraaQcy6IvcWJkWh1RsTDgUnk5C3PbgHsgLxVm
5A==
-----END ENCRYPTED PRIVATE KEY-----
```

Same format as the RSA private key

Inspecting the DSA key

Same command used for inspecting an RSA key:

```
openssl pkey -in giuperPrivateDSA.pem -text -noout
```



```
Private-Key: (2048 bit)
priv:
    30:8d:8c:fd:4d:86:94:85:9d:df:d4:87:ca:e4:4f:
    25:18:b8:45:69:4c:43:40:34:b1:34:37:e6:df:81:
    da:9e
pub:
    0d:a6:f6:55:2f:66:3d:31:de:5f:e6:60:e2:4b:e5:
    6b:ab:e4:04:a0:ee:b2:ce:dd:a7:a8:57:d7:1a:b7:
    34:02:ec:f1:47:28:5e:f7:99:71:9c:fe:9d:9b:db:
    ae:3d:37:4e:cd:71:32:e9:32:71:26:50:61:be:b0:
    8b:47:f3:fe:6a:9f:17:55:74:78:40:91:71:df:c4:
    8c:d4:90:b3:12:00:c8:dd:c1:c1:19:bc:8a:09:9b:
    18:95:e2:ba:32:7c:e0:23:c9:0d:60:a3:e9:38:25:
    eb:44:75:b2:b8:ff:fa:5e:11:6e:f6:bc:0d:87:76:
    0c:9b:95:a5:70:8a:12:59:59:c1:80:09:8f:72:71:
    1c:9e:34:5c:23:bb:54:ce:09:22:d2:2f:57:19:6f:
    ad:da:81:93:a1:23:6a:50:6a:92:b0:47:97:96:8e:
    6c:4e:2a:c8:5c:37:c5:49:32:d3:4e:00:c7:64:2d:
    ee:9a:e0:6d:4a:5e:e7:10:87:ff:82:71:0a:58:d9:
    7c:bd:ac:b3:7e:a2:bb:65:32:3b:ab:fd:82:5b:9c:
    0c:c4:75:2e:4a:de:73:7e:8d:6b:b1:81:d2:6d:c3:
    a0:2f:90:0f:84:b3:f0:cb:b4:54:22:25:d6:b4:68:
    dc:cf:88:15:c8:f1:ee:4e:ef:0f:ea:4e:8f:70:d8:
    9d
P:
    00:ae:4d:89:ab:2b:6d:e7:a8:04:b1:18:15:56:e3:
    21:89:5a:34:5f:94:0c:f1:b3:fe:76:9e:02:57:be:
    b1:21:02:d1:34:b2:48:29:56:80:7a:2c:66:f1:5b:
    f0:63:3f:71:b2:93:5d:a6:3a:ad:35:c5:03:57:d5:
```

Extracting the public key from the private key

```
openssl dsa -in giuperPrivateDSA.pem -out giuperPublicDSA.pem -pubout
```

How to sign and verify using DSA

- Sign:

  -
    ```
    openssl dgst -sha256 -sign giuperPrivateDSA.pem -out
    signatureFile documentFile.txt
    ```

- Verify:

  -
    ```
    openssl dgst -sha256 -verify giuperPublicDSA.pem
    -signature signatureFile documentFile.txt
    ```

Generating Keys for ECDSA

A two-step process

- First we generate the parameters

  ```
  openssl genpkey -genparam -algorithm EC
  ```

  options:

  - specify the curve: `-pkeyopt ec_paramgen_curve:  secp256k1`
  - the file containg the parameters: `-out`

- Then each user generates his/her pair of keys

  ```
  openssl genpkey
  ```

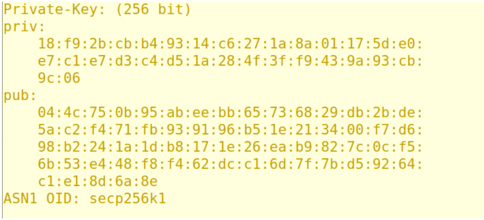
  options:

  - the armor algorithm: `-aes128`
  - the file with the parameters `-paramfile`
  - the file that will contain the private key `-out`

Inspecting the ECDSA key

Same command used for inspecting an RSA key:

```
openssl pkey -in giuperPrivateECDSA.pem -text -noout
```



```
Private-Key: (256 bit)
priv:
    18:f9:2b:cb:b4:93:14:c6:27:1a:8a:01:17:5d:e0:
    e7:c1:e7:d3:c4:d5:1a:28:4f:3f:f9:43:9a:93:cb:
    9c:06
pub:
    04:4c:75:0b:95:ab:ee:bb:65:73:68:29:db:2b:de:
    5a:c2:f4:71:fb:93:91:96:b5:1e:21:34:00:f7:d6:
    98:b2:24:1a:1d:b8:17:1e:26:ea:b9:82:7c:0c:f5:
    6b:53:e4:48:f8:f4:62:dc:c1:6d:7f:7b:d5:92:64:
    c1:e1:8d:6a:8e
ASN1 OID: secp256k1
```

399 bytes vs 1194 bytes (DSA) vs 1874 bytes (RSA)

How to sign and verify using ECDSA

- Sign:
  - `openssl dgst -sha256 -sign giuperPrivateECDSA.pem -out signatureFile documentFile.txt`

- Verify:
  - `openssl dgst -sha256 -verify giuperPublicECDSA.pem -signature signatureFile documentFile.txt`