Generate OpenSSL RSA Key Pair from the Command Line

Frank Rietta — 2012-01-27 (Last Updated: 2019-10-22)

While **Encrypting a File with a Password from the Command Line using OpenSSL** is very useful in its own right, the *real power* of the OpenSSL library is its ability to support the use of public key cryptograph for encrypting or validating data in an unattended manner (where the password is not required to encrypt) is done with public keys.

The Commands to Run

Generate a 2048 bit RSA Key

You can generate a public and private RSA key pair like this:

```
openssl genrsa -des3 -out private.pem 2048
```

That generates a 2048-bit RSA key pair, encrypts them with a password you provide and writes them to a file. You need to next extract the public key file. You will use this, for instance, on your web server to encrypt content so that it can only be read with the private key.

Export the RSA Public Key to a File

This is a command that is

```
openssl rsa -in private.pem -outform PEM -pubout -out public.pem
```

The `-pubout` flag is **really important**. Be sure to include it.

Next open the `public.pem` and ensure that it starts with `-----BEGIN PUBLIC KEY-----`. This is how you know that this file is the public key of the pair and not a private key.

To check the file from the command line you can use the `less` command, like this:

```
less public.pem
```

Do Not Run This, it Exports the Private Key

A previous version of the post gave this example in error.

```
openssl rsa -in private.pem -out private_unencrypted.pem -outform PEM
```

The error is that the `-pubout` was dropped from the end of the command. That changes the meaning of the command from that of exporting the public key to exporting the private key outside of its encrypted wrapper. Inspecting the output file, in this case `private_unencrypted.pem` clearly shows that the key is a RSA private key as it starts with `-----BEGIN RSA PRIVATE KEY-----`.

Visually Inspect Your Key Files

It is important to visually inspect you private and public key files to make sure that they are what you expect. OpenSSL will clearly explain the nature of the key block with a `-----BEGIN RSA PRIVATE KEY-----` or `-----BEGIN PUBLIC KEY-----`.

You can use less to inspect each of your two files in turn:

- `less private.pem` to verify that it starts with a `-----BEGIN RSA PRIVATE KEY-----`
- `less public.pem` to verify that it starts with a `-----BEGIN PUBLIC KEY-----`

The next section shows a full example of what each key file should look like.

The Generated Key Files

The generated files are base64-encoded encryption keys in plain text format. If you select a password for your private key, its file will be encrypted with your password. Be sure to remember this password or the key pair becomes useless.

The private.pem file looks something like this:

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,32495A90F3FF199D
lrMAsSjjkKiRxGdgR8p5kZJj0AFgdWYa3OT2snIXnN5+/p7j13PSkseUcrAFyokc
V9pgeDfitAhb9lpdjxjjuxRcuQjBfmNVLPF9MFyNOvhrprGNukUh/12oSKO9dFEt
s39F/2h6Ld5IQrGt3gZaBB1aGO+tw3ill1VBy2zGPIDeuSz6DS3GG/oQ2gLSSMP4
OVfQ32Oajo496iHRkdIh/7Hho7BNzMYr1GxrYTcE9/Znr6xgeSdNT37CCeCH8cmP
aEAUgSMTeIMVSpILwkKeNvBURic1EWaqXRgPRIWK0vNyOCs/+jNoFISnV4pu1ROF
92vayHDNSVw9wHcdSQ75XSE4Msawqv5U1iI7e2lD64uo1qhmJdrPcXDJQCiDbh+F
hQhF+wAoLRvMNwwhg+LttL8vXqMDQl3olsWSvWPs6b/MZpB0qwd1bklzA6P+PeAU
sfOvTqi9edIOfKqvXqTXEhBP8qC7ZtOKLGnryZb7W04SSVrNtuJUFRcLiqu+w/F/
MSxGSGalYpzIZ1B5HLQqISgWMXdbt39uMeeooeZjkuI3VIllFjtybecjPR9ZYQPt
FFEP1XqNXjLFmGh84TXtvGLWretWM1OZmN8UKKUeATqrr7zuh5AYGAIbXd8BvweL
```

```
Pigl9ei0hTculPqohvkoc5x1srPBvzHrirGlxOYjW3fc4kDgZpy+6ik5k5g7JWQD
lbXCRz3HGazgUPeiwUr06a52vhgT7QuNIUZqdHb4IfCYs2pQTLHzQjAqvVk1mm2D
kh4myIcTtf69BFcu/Wuptm3NaKd1nwk1squR6psvcTXOWII81pstnxNYkrokx4r2
7YVllNruOD+cMDNZbIG2CwT6V9ukIS8tl9EJp8eyb0a1uAEc22BNOjYHPF50beWF
ukf3uc0SA+G3zhmXCM5sMf5OxVjKr5jgcir7kySY5KbmG71omYhczgr4H0qgxYo9
Zyj2wMKrTHLfFOpd4OOEun9Gi3srqlKZep7Hj7gNyUwZu1qiBvElmBVmp0HJxT0N
mktuaVbaFgBsTS0/us1EqWvCA4REh1Ut/NoA9oG3JFt0lGDstTw1j+orDmIHOmSu
7FKYzr0uCz14AkLMSOixdPD1F0YyED1NMVnRVXw77HiAFGmb0CDi2KEg70pEKpn3
ksa8oe0MQi6oEwlMsAxVTXOB1wblTBuSBeaECzTzWE+/DHF+QQfQi8kAjjSdmmMJ
yN+shdBWHYRGYnxRkTatONhcDBIY7sZV7wolYHz/rf7dpYUZf37vdQnYV8FpO1um
Ya0GslyRJ5GqMBfDS1cQKne+FvVHxEE2YqEGBcOYhx/JI2soE8aA8W4XffN+DoEy
ZkinJ/+BOwJ/zUI9GZtwB4JXqbNEE+j7r7/fJO9KxfPp4MPK4YWu0H0EUWONpVwe
TWtbRhQUCOe4PVSC/Vv1pstvMD/D+E/0L4GQNHxr+xyFxuvILty5lvFTxoAVYpqD
u8gNhk3NWefTrlSkhY4N+tPP6o7E4t3y40nOA/d9qaqiid+lYcIDB0cJTpZvgeeQ
ijohxY3PHruU4vVZa37ITQnco9az6lsy18vbU0bOyK2fEZ2R9XVO8fH11jiV8oGH
-----END RSA PRIVATE KEY-----
```

The public key, public.pem, file looks like:

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAxzYuc22QSst/dS7geYYK
5l5kLxU0tayNdixkEQ17ix+CUcUbKIsnyftZxaCYT46rQtXgCaYRdJcbB3hmyrOa
vkhTpX79xJZnQmfuamMbZBqitvscxW9zRR9tBUL6vdi/0rpoUwPMEh8+Bw7CgYR0
FK0DhWYBNDfe9HKcyZEv3max8Cdq18htxjEsdYO0iwzhtKRXomBWTdhD5ykd/fAC
VTr4+KEY+IeLvubHVmLUhbE5NgWXxrRpGasDqzKhCTmsa2Ysf712rl57SlH0Wz/M
r3F7aM9YpErzeYLrl0GhQr9BVJxOvXcVd4kmY+XkiCcrkyS1cnghnllh+LCwQu1s
YwIDAQAB
-----END PUBLIC KEY-----
```

Protecting Your Keys

Depending on the nature of the information you will protect, it's important to keep the private key backed up and secret. The public key can be distributed anywhere or embedded in your web application scripts, such as in your PHP, Ruby, or other scripts. **Again, backup your keys!**

Remember, if the key goes away the data encrypted to it is gone. Keeping a printed copy of the key material in a sealed envelope in a bank safety deposit box is a good way to protect important keys against loss due to fire or hard drive failure.