Public Key Encryption and Digital Signatures using OpenSSL

By Steven Gordon on Sat, 07/04/2012 - 5:22pm

I recently gave students a homework task to get familiar with OpenSSL [3] as well as understand the use of public/private keys in public key cryptography (last year I gave same different tasks using certificates - see the steps [4]. The tasks for the student (sender in the notes below) were to:

- Create a RSA public/private key pair
- View and understand the parameters in the key pair
- Sign a message using their private key
- Encrypt a message using the recipients (my) public key
- "Send" the signature and ciphertext to the recipient (me)

Then I decrypted the ciphertext and verified the signature. Of course I also had to create my own key pair and make the public key available to the sender.

The steps are shown below, first in a screencast where I provide some explanation of the options and steps, and second in text form (with little explanation) that you can view and copy and paste if needed. Note that although the steps used in both outputs are the same, the actual values differ (i.e. the output listed below is from a different set of keys than used in the screencast).

1. Steps Performed by Sender

To generate the private (and public key):

```
$ openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:2048 -pkeyopt rsa_keygen_pul
$ cat privkey-ID.pem
-----BEGIN PRIVATE KEY-----
```

```
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQCoXEAmbAuh9Nks
xtjIqgW8+MjaoRLWIKOpr54E7XcpzMSlNZggPBp0sLjfgvNFBPP7BrQms3qigwow
krML/fdwSFybigmuTCyJS/UIn3J5s70vUSpQ9M8oAU+6lvRdiByqR0zBnnWdR9B8
wW2/jM2Ng3yq51S6qR6LUs92jEzYATz1df8z+qcUL+navmOSLdA110qQpbKjEjI1
esJIkqrKlQiu1N0TQbexC9dNwtI79G79UR+YOR8CWJyYy/ZPeUrsr1mcSGL7facW
/aG2hh85/XdICm2PWgRySUu0M2rHdxL+AMukauYnlw4gddTO0cmUNyxKrVr5aQBP
hZxKtFV5AgEDAoIBAHA9gBmdXRajO3MvOzBxWSil2zxrYeQVwnEfvq3zpMaIgxjO
ZWrSvE3LJepXTNit9/yvIsR3pxcCBssMd11T+kra6GexW8mIHbDdTgW/oaZ303Tg
xuCjNMVWNScPTZOwExwviIEUTmjaiv3WSSpd3l5XqHHvjdHGFFzh36RdiI//vcSX
VHC76AkhkJ13aDEIUSQPMfE0OmI4dgK2sxH8BXAmAgc7YOksLF4t+tjaEoeUFQWP
SwFiGgVaU3wtmv1DoSwbAKSWs/9hDg3vgN8AFku3HCdBkpmpp2CYqoBWFDfUNW2q
TtB7IU2fwUOtoqiW8CegqVNf+X+KWT85mb1NnqMCgYEA3z2IhWyENYsHRrfbpISR
q3y5l5sgFM1ofRbPA5AZbZANY48jFPSeuKWJ1HhhZpwai+dcKf5R2w5V/4vpKqec
wFFGkXiOshkzty/67A75Uww/iewff0nj8ZwG7oLYl2PHu7iyyHiwbTj7N21Rapq+
iUHpd4RBpiOPoad4lD+CDWcCgYEAwREKex5clXt2SjavosQPqwMG6Au3RkJVBBqZ
sh1/NRJOohTYtsDgvH49CpAaT9R7w42eBRfUHOv7H9KeYyv3GNlARyzXouM4WtIb
dFkMqrwrQyEIkl73l8VdXXDZtQ/xByDOjPMBxvosNM2f9jcw2BbctslbvpaJ2Mk2
oW892h8CgYEAlNOwWPMCzlyvhHqSba22clMmZRIVYzOa/g80rQq7nmAI7QoXY02/
JcOxOFBA7xK8XUToG/7hPLQ5VQfwxxpogDYvC6W0drt3z3VR8rSmN11/sUgU/4aX
9mgEnwHlukKFJ9B3MFB1niX8z542RxHUW4FGT62BGW0Ka8T7DX+sCO8CgYEAgLYG
/L7oY6ekMXnKbIK1HKyvRV0k2YGOArxmdr5Uzgw0bA3lzytAfal+Bwq8NThSgl5p
WLqNaJ1SFTcUQh1PZeYq2h3lF0IlkeFnouYIcdLHghYFtun6ZS4+Pks7zgqgr2s0
XfdWhKbIIzO/+XogkA89zzDn1GRb5dt5wPTT5r8CgYEA29235n/Hw7wzOJyao6nO
3rjCZon4/V2G800VJF5hhAqCX5KDLd0KIMbaHaxsjW+n79CqZSUz3kZtpSXBXRJ7
SIXoCYljaoxdJ6SkVED6uFmcZ+3iwioxXzpIFIW0ZZj5S/WgBkPsioAJ6Cp5S8zh
BFB15UA+JWFH2SRabjXf0+4=
-----END PRIVATE KEY-----
```

The private key is encoded with Base64. To view the values:

```
$ openssl pkey -in privkey-ID.pem -text
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBKcwggSjAgEAAoIBAQCoXEAmbAuh9Nks
xtjIqgW8+MjaoRLWIKOpr54E7XcpzMSlNZggPBp0sLjfgvNFBPP7BrQms3qigwow
krML/fdwSFybigmuTCyJS/UIn3J5s70vUSpQ9M8oAU+6lvRdiByqR0zBnnWdR9B8
wW2/jM2Ng3yq51S6qR6LUs92jEzYATz1df8z+qcUL+navmOSLdA110qQpbKjEjI1
esJIkqrKlQiu1N0TQbexC9dNwtI79G79UR+YOR8CWJyYy/ZPeUrsr1mcSGL7facW
/aG2hh85/XdICm2PWgRySUu0M2rHdxL+AMukauYnlw4gddTO0cmUNyxKrVr5aQBP
hZxKtFV5AgEDAoIBAHA9gBmdXRajO3MvOzBxWSil2zxrYeQVwnEfvq3zpMaIgxjO
ZWrSvE3LJepXTNit9/yvIsR3pxcCBssMd11T+kra6GexW8mIHbDdTgW/oaZ303Tg
xuCjNMVWNScPTZOwExwviIEUTmjaiv3WSSpd3l5XqHHvjdHGFFzh36RdiI//vcSX
VHC76AkhkJ13aDEIUSQPMfE0OmI4dgK2sxH8BXAmAgc7YOksLF4t+tjaEoeUFQWP
SwFiGgVaU3wtmv1DoSwbAKSWs/9hDg3vgN8AFku3HCdBkpmpp2CYqoBWFDfUNW2q
TtB7IU2fwUOtoqiW8CegqVNf+X+KWT85mb1NnqMCgYEA3z2IhWyENYsHRrfbpISR
q3y5l5sgFM1ofRbPA5AZbZANY48jFPSeuKWJ1HhhZpwai+dcKf5R2w5V/4vpKqec
wFFGkXiOshkzty/67A75Uww/iewff0nj8ZwG7oLYl2PHu7iyyHiwbTj7N21Rapq+
iUHpd4RBpiOPoad4lD+CDWcCgYEAwREKex5clXt2SjavosQPqwMG6Au3RkJVBBqZ
sh1/NRJOohTYtsDgvH49CpAaT9R7w42eBRfUHOv7H9KeYyv3GNlARyzXouM4WtIb
dFkMqrwrQyEIkl73l8VdXXDZtQ/xByDOjPMBxvosNM2f9jcw2BbctslbvpaJ2Mk2
oW892h8CgYEAlNOwWPMCzlyvhHqSba22clMmZRIVYzOa/g80rQq7nmAI7QoXY02/
JcOxOFBA7xK8XUToG/7hPLQ5VQfwxxpogDYvC6W0drt3z3VR8rSmN11/sUgU/4aX
9mgEnwHlukKFJ9B3MFB1niX8z542RxHUW4FGT62BGW0Ka8T7DX+sCO8CgYEAgLYG
/L7oY6ekMXnKbIK1HKyvRV0k2YGOArxmdr5Uzgw0bA3lzytAfal+Bwq8NThSgl5p
WLqNaJ1SFTcUQh1PZeYq2h3lF0IlkeFnouYIcdLHghYFtun6ZS4+Pks7zgqgr2s0
XfdWhKbIIzO/+XogkA89zzDn1GRb5dt5wPTT5r8CgYEA29235n/Hw7wzOJyao6nO
3rjCZon4/V2G800VJF5hhAqCX5KDLd0KIMbaHaxsjW+n79CqZSUz3kZtpSXBXRJ7
SIXoCYljaoxdJ6SkVED6uFmcZ+3iwioxXzpIFIW0ZZj5S/WgBkPsioAJ6Cp5S8zh
```

```
BFB15UA+JWFH2SRabjXf0+4=
-----END PRIVATE KEY-----
Private-Key: (2048 bit)
modulus:
    00:a8:5c:40:26:6c:0b:a1:f4:d9:2c:c6:d8:c8:aa:
    05:bc:f8:c8:da:a1:12:d6:20:a3:a9:af:9e:04:ed:
    77:29:cc:c4:a5:35:98:20:3c:1a:74:b0:b8:df:82:
    f3:45:04:f3:fb:06:b4:26:b3:7a:a2:83:0a:30:92:
    b3:0b:fd:f7:70:48:5c:9b:8a:09:ae:4c:2c:89:4b:
    f5:08:9f:72:79:b3:bd:2f:51:2a:50:f4:cf:28:01:
    4f:ba:96:f4:5d:88:1c:aa:47:4c:c1:9e:75:9d:47:
    d0:7c:c1:6d:bf:8c:cd:8d:83:7c:aa:e7:54:ba:a9:
    1e:8b:52:cf:76:8c:4c:d8:01:3c:f5:75:ff:33:fa:
    a7:14:2f:e9:da:be:63:92:2d:d0:35:d7:4a:90:a5:
    b2:a3:12:32:35:7a:c2:48:92:aa:ca:95:08:ae:d4:
    dd:13:41:b7:b1:0b:d7:4d:c2:d2:3b:f4:6e:fd:51:
    1f:98:39:1f:02:58:9c:98:cb:f6:4f:79:4a:ec:af:
    59:9c:48:62:fb:7d:a7:16:fd:a1:b6:86:1f:39:fd:
    77:48:0a:6d:8f:5a:04:72:49:4b:b4:33:6a:c7:77:
    12:fe:00:cb:a4:6a:e6:27:97:0e:20:75:d4:ce:d1:
    c9:94:37:2c:4a:ad:5a:f9:69:00:4f:85:9c:4a:b4:
    55:79
publicExponent: 3 (0x3)
privateExponent:
    70:3d:80:19:9d:5d:16:a3:3b:73:2f:3b:30:71:59:
    28:a5:db:3c:6b:61:e4:15:c2:71:1f:be:ad:f3:a4:
    c6:88:83:18:ce:65:6a:d2:bc:4d:cb:25:ea:57:4c:
    d8:ad:f7:fc:af:22:c4:77:a7:17:02:06:cb:0c:77:
    5d:53:fa:4a:da:e8:67:b1:5b:c9:88:1d:b0:dd:4e:
    05:bf:a1:a6:77:d3:74:e0:c6:e0:a3:34:c5:56:35:
    27:0f:4d:93:b0:13:1c:2f:88:81:14:4e:68:da:8a:
    fd:d6:49:2a:5d:de:5e:57:a8:71:ef:8d:d1:c6:14:
    5c:e1:df:a4:5d:88:8f:ff:bd:c4:97:54:70:bb:e8:
    09:21:90:9d:77:68:31:08:51:24:0f:31:f1:34:3a:
    62:38:76:02:b6:b3:11:fc:05:70:26:02:07:3b:60:
    e9:2c:2c:5e:2d:fa:d8:da:12:87:94:15:05:8f:4b:
    01:62:1a:05:5a:53:7c:2d:9a:fd:43:a1:2c:1b:00:
    a4:96:b3:ff:61:0e:0d:ef:80:df:00:16:4b:b7:1c:
    27:41:92:99:a9:a7:60:98:aa:80:56:14:37:d4:35:
    6d:aa:4e:d0:7b:21:4d:9f:c1:43:ad:a2:a8:96:f0:
    27:a0:a9:53:5f:f9:7f:8a:59:3f:39:99:bd:4d:9e:
    a3
prime1:
    00:df:3d:88:85:6c:84:35:8b:07:46:b7:db:a4:84:
    91:ab:7c:b9:97:9b:20:14:cd:68:7d:16:cf:03:90:
    19:6d:90:0d:63:8f:23:14:f4:9e:b8:a5:89:d4:78:
    61:66:9c:1a:8b:e7:5c:29:fe:51:db:0e:55:ff:8b:
    e9:2a:a7:9c:c0:51:46:91:78:8e:b2:19:33:b7:2f:
    fa:ec:0e:f9:53:0c:3f:89:ec:1f:7f:49:e3:f1:9c:
    06:ee:82:d8:97:63:c7:bb:b8:b2:c8:78:b0:6d:38:
    fb:37:6d:51:6a:9a:be:89:41:e9:77:84:41:a6:23:
    8f:a1:a7:78:94:3f:82:0d:67
prime2:
    00:c1:11:0a:7b:1e:5c:95:7b:76:4a:36:af:a2:c4:
    0f:ab:03:06:e8:0b:b7:46:42:55:04:1a:99:b2:1d:
    7f:35:12:4e:a2:14:d8:b6:c0:e0:bc:7e:3d:0a:90:
    1a:4f:d4:7b:c3:8d:9e:05:17:d4:1c:eb:fb:1f:d2:
    9e:63:2b:f7:18:d9:40:47:2c:d7:a2:e3:38:5a:d2:
```

```
    1b:74:59:0c:aa:bc:2b:43:21:08:92:5e:f7:97:c5:
    5d:5d:70:d9:b5:0f:f1:07:20:ce:8c:f3:01:c6:fa:
    2c:34:cd:9f:f6:37:30:d8:16:dc:b6:c9:5b:be:96:
    89:d8:c9:36:a1:6f:3d:da:1f
exponent1:
    00:94:d3:b0:58:f3:02:ce:5c:af:84:7a:92:6d:ad:
    b6:72:53:26:65:12:15:63:33:9a:fe:0f:34:ad:0a:
    bb:9e:60:08:ed:0a:17:63:4d:bf:25:c3:b1:38:50:
    40:ef:12:bc:5d:44:e8:1b:fe:e1:3c:b4:39:55:07:
    f0:c7:1a:68:80:36:2f:0b:a5:b4:76:bb:77:cf:75:
    51:f2:b4:a6:37:5d:7f:b1:48:14:ff:86:97:f6:68:
    04:9f:01:e5:ba:42:85:27:d0:77:30:50:75:9e:25:
    fc:cf:9e:36:47:11:d4:5b:81:46:4f:ad:81:19:6d:
    0a:6b:c4:fb:0d:7f:ac:08:ef
exponent2:
    00:80:b6:06:fc:be:e8:63:a7:a4:31:79:ca:6c:82:
    b5:1c:ac:af:45:5d:24:d9:81:8e:02:bc:66:76:be:
    54:ce:0c:34:6c:0d:e5:cf:2b:40:7d:a9:7e:07:0a:
    bc:35:38:52:82:5e:69:58:ba:8d:68:9d:52:15:37:
    14:42:1d:4f:65:e6:2a:da:1d:e5:17:42:25:91:e1:
    67:a2:e6:08:71:d2:c7:82:16:05:b6:e9:fa:65:2e:
    3e:3e:4b:3b:ce:0a:a0:af:6b:34:5d:f7:56:84:a6:
    c8:23:33:bf:f9:7a:20:90:0f:3d:cf:30:e7:d4:64:
    5b:e5:db:79:c0:f4:d3:e6:bf
coefficient:
    00:db:dd:b7:e6:7f:c7:c3:bc:33:38:9c:9a:a3:a9:
    ce:de:b8:c2:66:89:f8:fd:5d:86:f3:4d:15:24:5e:
    61:84:0a:82:5f:92:83:2d:dd:0a:20:c6:da:1d:ac:
    6c:8d:6f:a7:ef:d0:aa:65:25:33:de:46:6d:a5:25:
    c1:5d:12:7b:48:85:e8:09:89:63:6a:8c:5d:27:a4:
    a4:54:40:fa:b8:59:9c:67:ed:e2:c2:2a:31:5f:3a:
    48:14:85:b4:65:98:f9:4b:f5:a0:06:43:ec:8a:80:
    09:e8:2a:79:4b:cc:e1:04:50:75:e5:40:3e:25:61:
    47:d9:24:5a:6e:35:df:d3:ee
```

To output just the public key to a file:

```
$ openssl pkey -in privkey-ID.pem -out pubkey-ID.pem -pubout
$ cat pubkey-ID.pem
-----BEGIN PUBLIC KEY-----
MIIBIDANBgkqhkiG9w0BAQEFAAOCAQ0AMIIBCAKCAQEAqFxAJmwLofTZLMbYyKoF
vPjI2qES1iCjqa+eBO13KczEpTWYIDwadLC434LzRQTz+wa0JrN6ooMKMJKzC/33
cEhcm4oJrkwsiUv1CJ9yebO9L1EqUPTPKAFPupb0XYgcqkdMwZ51nUfQfMFtv4zN
jYN8qudUuqkei1LPdoxM2AE89XX/M/qnFC/p2r5jki3QNddKkKWyoxIyNXrCSJKq
ypUIrtTdE0G3sQvXTcLSO/Ru/VEfmDkfAlicmMv2T3lK7K9ZnEhi+32nFv2htoYf
Of13SAptj1oEcklLtDNqx3cS/gDLpGrmJ5cOIHXUztHJlDcsSq1a+WkAT4WcSrRV
eQIBAw==
-----END PUBLIC KEY-----
```

Check by looking at the invidual values:

```
$ openssl pkey -in pubkey-ID.pem -pubin -text
-----BEGIN PUBLIC KEY-----
MIIBIDANBgkqhkiG9w0BAQEFAAOCAQ0AMIIBCAKCAQEAqFxAJmwLofTZLMbYyKoF
vPjI2qES1iCjqa+eBO13KczEpTWYIDwadLC434LzRQTz+wa0JrN6ooMKMJKzC/33
```

```
cEhcm4oJrkwsiUv1CJ9yebO9L1EqUPTPKAFPupb0XYgcqkdMwZ51nUfQfMFtv4zN
jYN8qudUuqkei1LPdoxM2AE89XX/M/qnFC/p2r5jki3QNddKkKWyoxIyNXrCSJKq
ypUIrtTdE0G3sQvXTcLSO/Ru/VEfmDkfAlicmMv2T3lK7K9ZnEhi+32nFv2htoYf
Of13SAptj1oEcklLtDNqx3cS/gDLpGrmJ5cOIHXUztHJlDcsSq1a+WkAT4WcSrRV
eQIBAw==
-----END PUBLIC KEY-----
Public-Key: (2048 bit)
Modulus:
    00:a8:5c:40:26:6c:0b:a1:f4:d9:2c:c6:d8:c8:aa:
    05:bc:f8:c8:da:a1:12:d6:20:a3:a9:af:9e:04:ed:
    77:29:cc:c4:a5:35:98:20:3c:1a:74:b0:b8:df:82:
    f3:45:04:f3:fb:06:b4:26:b3:7a:a2:83:0a:30:92:
    b3:0b:fd:f7:70:48:5c:9b:8a:09:ae:4c:2c:89:4b:
    f5:08:9f:72:79:b3:bd:2f:51:2a:50:f4:cf:28:01:
    4f:ba:96:f4:5d:88:1c:aa:47:4c:c1:9e:75:9d:47:
    d0:7c:c1:6d:bf:8c:cd:8d:83:7c:aa:e7:54:ba:a9:
    1e:8b:52:cf:76:8c:4c:d8:01:3c:f5:75:ff:33:fa:
    a7:14:2f:e9:da:be:63:92:2d:d0:35:d7:4a:90:a5:
    b2:a3:12:32:35:7a:c2:48:92:aa:ca:95:08:ae:d4:
    dd:13:41:b7:b1:0b:d7:4d:c2:d2:3b:f4:6e:fd:51:
    1f:98:39:1f:02:58:9c:98:cb:f6:4f:79:4a:ec:af:
    59:9c:48:62:fb:7d:a7:16:fd:a1:b6:86:1f:39:fd:
    77:48:0a:6d:8f:5a:04:72:49:4b:b4:33:6a:c7:77:
    12:fe:00:cb:a4:6a:e6:27:97:0e:20:75:d4:ce:d1:
    c9:94:37:2c:4a:ad:5a:f9:69:00:4f:85:9c:4a:b4:
    55:79
Exponent: 3 (0x3)
```

Create a text file:

```
$ cat message-ID.txt
This is my example message.
```

To sign the message you need to calculate its hash and then encrypt that hash using your private key. To create a hash of a message (without encrypting):

```
$ openssl dgst -sha1 message-ID.txt
SHA1(message-ID.txt)= 064774b2fb550d8c1d7d39fa5ac5685e2f8b1ca6
```

OpenSSL has an option to calculate the hash and then sign it:

```
$ openssl dgst -sha1 -sign privkey-ID.pem -out sign-ID.bin message-ID.txt
$ ls -l
total 16
-rw-r--r-- 1 sgordon users   28 2012-03-04 15:14 message-ID.txt
-rw-r--r-- 1 sgordon users 1704 2012-03-04 14:58 privkey-ID.pem
-rw-r--r-- 1 sgordon users  451 2012-03-04 15:08 pubkey-ID.pem
-rw-r--r-- 1 sgordon users  256 2012-03-04 15:20 sign-ID.bin
```

To encrypt the message using RSA, use the recipients public key:

```
$ openssl pkeyutl -encrypt -in message.txt -pubin -inkey pubkey-Steve.pem -out cipher
```

2. Steps Performed by Receiver

The public key was generated and made available to the sender:

```
$ openssl genpkey -algorithm RSA -pkeyopt rsa_keygen_bits:2048 -pkeyopt rsa_keygen_pul
$ openssl pkey -in privkey-Steve.pem -out pubkey-Steve.pem -pubout
```

To decrypt the received ciphertext:

```
$ openssl pkeyutl -decrypt -in ciphertext-ID.bin -inkey privkey-Steve.pem -out receive
$ cat received-ID.txt
This is my example message.
```

To verify the signature of a message:

```
$ openssl dgst -sha1 -verify pubkey-ID.pem -signature sign-ID.bin received-ID.txt
Verified OK
```

Interest: Ubuntu Linux [5]
OpenSSL [6]
Topic: Security [7]
Content: Howto [8]

Source URL: http://sandilands.info/sgordon/public-key-encryption-and-digital-signatures-using-openssl

Links:
[1] http://sandilands.info/sgordon/public-key-encryption-and-digital-signatures-using-openssl
[2] http://sandilands.info/sgordon/user/2
[3] http://www.openssl.com/
[4] http://sandilands.info/sgordon/key-generation-and-encryption-examples-using-openssl
[5] http://sandilands.info/sgordon/taxonomy/term/302
[6] http://sandilands.info/sgordon/taxonomy/term/338
[7] http://sandilands.info/sgordon/taxonomy/term/116
[8] http://sandilands.info/sgordon/taxonomy/term/212