

RSA con OpenSSL in Linux

L'encryption (**Cifratura**) e il Decryption (**Decifratura**) con algoritmi **a chiave asimmetrica** prevede l'uso di **due chiavi**, una **pubblica** nota a tutti i partecipanti e una **privata** da custodire con cura.

OpenSSL è un toolkit robusto, di livello commerciale e completo di funzionalità per la crittografia generica e la comunicazione sicura.

Per lavorare con OpenSSL il tool deve essere installato. Per verificare che il tool sia correttamente installato digitare il comando seguente:

```
openssl version
OpenSSL 3.0.2 15 Mar 2022 (Library: OpenSSL 3.0.2 15 Mar 2022)
```

Esercitazione 1. Ipotizziamo una comunicazione tra Alice e Bob.

1. Creare una cartella e nominarla "RSA":

```
(psygnosys@psygnosys)-[~]
$ mkdir RSA
```

2. Spostarsi in "RSA" e creare il file "clear.txt":

```
(psygnosys@psygnosys)-[~]
$ cd RSA

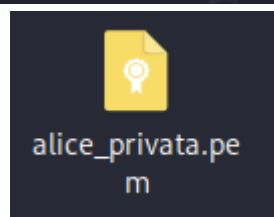
(psygnosys@psygnosys)-[~/RSA]
$ touch clear.txt
```

3. Scrivere la frase "ciao questo è un file in chiaro da rendere segreto" nel file "clear.txt":

```
(psygnosys@psygnosys)-[~/RSA]
$ echo "Ciao questo è un file in chiaro da rendere segreto" > clear.txt
```

4. Alice genera la propria chiave privata con parametri di default a 2048 bit:

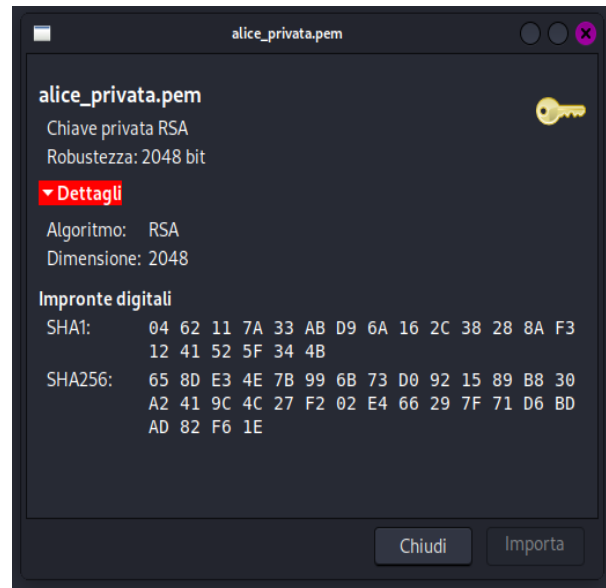
```
(psygnosys@psygnosys)-[~/RSA]
$ openssl genpkey -algorithm RSA -out alice_privata.pem
```



5. Visualizziamo il contenuto di "alice_privata.pem":

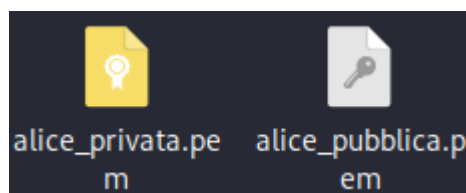
```
(psygnosys@psygnosys)-[~/RSA]
$ code alice_privata.pem
```

```
-----BEGIN PRIVATE KEY-----
MIIEvQIBADANBgkqhkiG9w0BAQEFAASCBCwggSjAgEAAoIBAQDEGL9tuPvrHOE
xb5XYjAcFvBDDA3I0RFCKs08aNuN86aP6nAxFCrm/eEcSsbFXW8HVIvBYBZHP4b
LVU/qR6JDYbfxmKwcGHnvFNWR6pj67NNBGHG00dKinWLV0DF0XLQsi8bKr48wtxb
Qpu0tv5ItgywGUGbS3vNsZdg4I/AP0DfMuYL7KehDuF9noIplxYQap5gVDPs3XLk
bltvqQ5FMBddKbxSGVqPwjMwinWCsgfLD56LzMs43DJU1Y/IWvSt8/v4uQYxXwXh
bRv1MYAuEeFBKIPAVptHc05hThZIHueNf/myJw4XWlpL6/64XtEubbXysYVQ11C
6xzFD0FLAgMBAAECggEAWqfRqHRYDNG7N9rZA3nLhvjdywk0CC4K3f16LsyFD6tf
ybGWe3INPwUkGbpCD7UBorBi1FloyAm8udDAL+ZGHoMH4sTNlUnHmQ0XLNV5Mqu
wgCwGN4Ht8Cp4a1La+hHvY5ac6M/fkSRbYkt8/zKMqAi3nhWma38Rpv+wRUfr75f
AY4hwOuqI2PQsQoYF8Y7cz5hkuDXTktYgfBvtz80BKtjScMtbwR1NK6jqodHU8Lh
LGA14b+c7iEfoANvo59jmDMS3KNNVTdrozjV75aQGVuXwDp7YvRbkhQXxQ9C1Tji
i9Huk7MRKw/1KZHkj8B6It0bXsS0gtZM4VzdSk20XQKBgQDnNc7KZm1AAEeggLCA
BRL2NcwpSaq/2C9KQCM0YXWtKxdZB73P7Di34djYAkY3P0uF0cSjv8rS9ixEMVCP
s1Cd60viEtrM0miIdI04uek05PeT176nr3FTltDekp2n0qfM0ZSNLfjBLmXuL0Km
KCm7if0V+aDGasy+yZoJr1QZ/QKBgQDZH6dL3wcgBIrrFXoNBx2FGPRSGJAUR+
o4iYQHwocGJZCg92cNIP9dbuziLD2jGEcXuzniFabbK4+1N7WUX+M8bxZ1h7+IO
U0XN/onlCp71jqk10eHdwAp27daRwD8CxcA1WqPLZsbG7f16U2V6rJvMv08ZupD
CSnys2Vm5wKBgH2LiCf7Ci0QgImf2zcIjdTUTXhtd0jImJUW0ocdRvQsZVcgjv7R
49JlPhu80Ge7ZMTbLH8nt3hp14uHw5jj40h7a2oslsZVEhHWR8CfonPoSE7KP0pM
p5qd7/iLYDP6fMcInkMyzUgmn45/9w7GDDMbKzsqexSCdX67dXsAywf5AoGAMnCS
GeBzLkV/WCqEQfPqslf3spn1VWoox2F2zaPNoogXhwx+kMfbJxi1tjLUwU+a010F
/dP3iaa5SW8j/PuOhcAmlSQWiN8KnBtaVx2v+Ua7YP6Qbq4Qu8xwYgSk2V39B5Cn
+miaBcMpiMwPj9Z598LLwPo9hR0jd4+IYTtpLcCgYEAtIbHK1XULoSfEm4GE4Jc
YJAI7ToxLsmihUXuR1USHQvnyaIy2Zhl0g+0lFqWzFqkRgBGezQotv16N/guKmj
qXsboic2GdotDdfEs3BbwcNoGyEkH8GcVCBfyj5v5/4Av7vAbJc6hJHvVh+CoU5x
Ext98xdiRy0EsD7UoC+ydBQ=
-----END PRIVATE KEY-----
```



6. Generiamo la chiave pubblica da quella privata (sono matematicamente correlate):

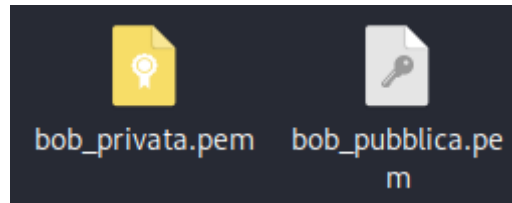
```
(psygnosys@psygnosys)-[~/RSA]
$ openssl rsa -in alice_privata.pem -pubout > alice_pubblica.pem
writing RSA key
```



7. Analizziamo la chiave pubblica generata:

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAACCAQ8AMIIBCgKCAQEAxBky/bbj76xzhMW+V2Iw
HBbwQwwNyNERQirDvGjbjf0mj+pwMXwq5v3hHErGxV1vB1SFW2WAWRz+G5VVP6q+
iQ2G38ZpMHBh57xTVkeqY+uzTQRhxjtHSop1pVaAxdFy0LIvGyq+PMLcW0Kbjrb+
SLYMsB1Bm0t7zbGXYOCPwD9A3zLmJeynoQ7hfZ6CKZcWEGqeYFQ6Ut1y5G5bb6qu
RTAXXSm8Uhlaj8IzFop1grIH5Q+ei8zLONwyVNWPyFr0rfP7+LkGMV8F4W0b9TGA
LhHhQZCDwFabR3N0YU4WRyB7njX/5sic0F1paS+v+uF7RF6218rGFUNdQusc3w9B
SwIDAQAB
-----END PUBLIC KEY-----
```

8. Bob eseguirà i punti 4) e 6) ottenendo la propria chiave privata e pubblica:



9. Dopo che Alice e Bob si scambiano le chiavi pubbliche utilizzando un protocollo sicuro come ad esempio SSH, Alice (mittente) vuole garantire confidenzialità cifrando il messaggio con la chiave pubblica di Bob:

```
(psygnosys@psygnosys)-[~/RSA]
$ openssl pkeyutl -encrypt -pubin -inkey bob_pubblica.pem -in clear.txt | base64 > chypher.txt
```

- **pkeyutl**: abilita il tool per le chiavi pubbliche
- **-encrypt**: abilita il protocollo di encryption
- **-pubin**: specifica che in ingresso c'è una chiave pubblica
- **-inkey**: specifica quale chiave utilizzare
- **base64**: encoding in base64

10. Verifichiamo il contenuto di chypher.txt:

```
chypher.txt X
home > psygnosys > RSA > chypher.txt
1 |ThDhxvLj7fbN9x4Dk6oh051IQvhYno/Xn3gpXQhBnvF/0FNSXp/pbLHoH7iVd70/Qw21tRw+3Fgr
2 2IIXBBwG9iDUM/bLZNtOCWYuLVDi/T9i8zGYpW15fdIw/rwNZ1cwrcFKHWPNMsmvUhZFI0NcGxsD
3 EsTJm6NYWnNAIfTJxv06hk50NuTWF4yhaPphqqtV04wgsfc94Jfq5+rrsZVbIS9pKyToGPbyr/k6
4 gsGESAUpgsF9FK4U7guUcUl/aWA0yQG6Ne/YJFkZ79Ku/ziXuR3BvkwDfF7v0CzP6fPZe5Yt0b3I
5 EW6WdsaH6sa3ZPC7+cCE4dV8p66sJ1dtcwzmEA==
```

11. Bob decifra il file con la propria chiave privata e ne visualizza il contenuto:

```
(psygnosys@psygnosys)-[~/RSA]
$ cat chypher.txt | base64 -d | openssl pkeyutl -decrypt -inkey bob_privata.pem -out clear_conf.txt

(psygnosys@psygnosys)-[~/RSA]
$ cat clear_conf.txt
Ciao questo è un file in chiaro da rendere segreto
```

12. Ipotizzando che John sia un attaccante e intercetti il messaggio, se prova a decifrare con le proprie chiavi:

16. Bob può decriptare il file con la propria chiave privata e con quella pubblica di Alice:

```
(psygnosys@psygnosys)-[~/RSA]
$ openssl pkeyutl -decrypt -inkey bob_privata.pem -in pagamento_enc.bin -out pagamento.txt
md5sum pagamento.txt > pagamento_hash.md5
openssl pkeyutl -verify -pubin -inkey alice_pubblica.pem -sigfile pagamento_signature.bin -in pagamento_hash.md5
Signature Verified Successfully
```