RSA in OpenSSL

Alfredo De Santis

Dipartimento di Informatica
Università di Salerno

ads@unisa.it

http://www.dia.unisa.it/professori/ads

Marzo 2017

Rappresentazione e codifica dei dati

- Chiavi e parametri utilizzati da OpenSSL si basano su ASN.1 e sono generalmente codificati in formato PEM o DER
- Abstract Syntax Notation One (ASN.1)

  Descrive le informazioni che saranno scambiate indipendentemente da come tali informazioni sono rappresentate su ognuno dei sistemi che comunicano
- Regole di codifica, che specificano come rappresentare i dati descritti mediante ASN.1
- PKCS: Standard che descrivono, mediante ASN.1, gli oggetti coinvolti in operazioni crittografiche

ASN.1

- Abstract Syntax Notation One (ASN.1)
- ASN.1 è un modo per descrivere i dati
  - Partendo da tipi primitivi e costruendo tipi più complessi
- ASN.1 specifica tipo e struttura dei dati
  - Tipo di valore
    - Intero, booleano, stringa di caratteri, etc.
  - Struttura
    - Contenimento, ordine, opzioni
- ASN.1 non specifica la codifica (rappresentazione) dei dati

ASN.1
(esempio: chiave privata RSA in PKCS#1)

```
RSAPrivateKey ::= SEQUENCE {
   version          Version,
   modulus          INTEGER,  -- n
   publicExponent   INTEGER,  -- e
   privateExponent  INTEGER,  -- d
   prime1           INTEGER,  -- p
   prime2           INTEGER,  -- q
   exponent1        INTEGER,  -- d mod (p-1)
   exponent2        INTEGER,  -- d mod (q-1)
   coefficient      INTEGER,  -- (inverse of q) mod p
   otherPrimeInfos  OtherPrimeInfos OPTIONAL
}

Version ::= INTEGER { two-prime(0), multi(1) }
   (CONSTRAINED BY {
      -- version must be multi if otherPrimeInfos present --
   })

OtherPrimeInfos ::= SEQUENCE SIZE(1..MAX) OF OtherPrimeInfo

OtherPrimeInfo ::= SEQUENCE {
   prime            INTEGER,  -- ri
   exponent         INTEGER,  -- di
   coefficient      INTEGER   -- ti
}
```

Regole di Codifica

- ASN.1 è strettamente relato ad un insieme di regole di codifica, che specificano come rappresentare i dati descritti mediante ASN.1

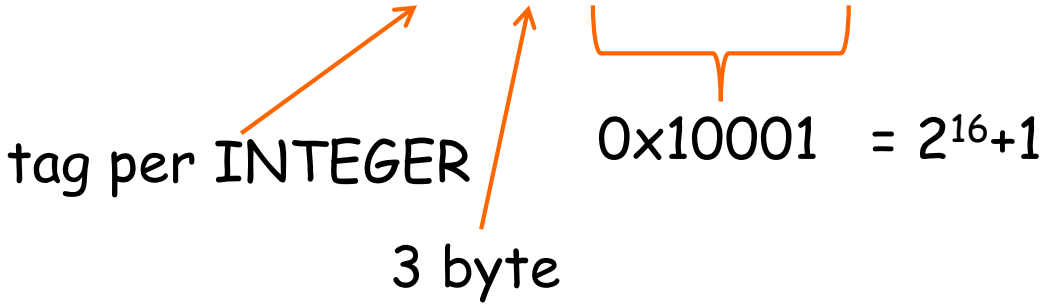
- Tali regole sono spesso caratterizzate da una specifica estensione (formato)
  - Basic Encoding Rules (BER)
  - Distinguished Encoding Rules (DER)
  - Canonical Encoding Rules (CER)
  - Etc.

- Progettate inizialmente per la trasmissione tramite comunicazione seriale
  - Dove i dati sono inviati un bit alla volta, in modo sequenziale

DER

- Distinguished Encoding Rules (DER)
  - Sottoinsieme di BER
  - Codifica in binario i dati descritti mediante ASN.1
  - Ogni file può contenere un solo oggetto codificato in DER

Esempio: `02 03 01 00 01`



DER vs. PEM

- Distinguished Encoding Rules (DER)
  - Sottoinsieme di BER
  - Codifica in binario i dati descritti mediante ASN.1
  - Ogni file può contenere un solo oggetto codificato in DER
- Privacy Enhanced Mail (PEM)
  - Formato di default usato da OpenSSL
  - Usato per codificare in formato testuale (Base64) oggetti codificati tramite DER
  - Può contenere commenti
  - Ogni file può contenere oggetti multipli codificati in PEM
    - Uno sotto l'altro
  - Sintassi

```
-----BEGIN keyword-----
Base 64 encoding of BER/DER
-----END keyword-----
```

DER vs. PEM

- Distinguished Encoding Rules (DER)
  - Sottoinsieme di BER
  - Codifica in binario i dati descritti mediante ASN.1
  - Ogni file può contenere un solo oggetto codificato in DER
- Privacy Enhanced Mail (PEM)
  - Formato di default usato da OpenSSL
  - Usato per codificare in formato testuale (Base64) oggetti codificati tramite DER
  - Può contenere comment
  - Ogni file può contenere
    - Uno sotto l'altro
  - Sintassi

PEM

keyword può essere una chiave pubblica, una chiave privata, etc

```
-----BEGIN keyword-----
Base 64 encoding of BER/DER
-----END keyword-----
```

DER vs. PEM

- Distinguished Encoding Rules (DER)
  - Sottoinsieme di BER
  - Codifica in binario i dati descritti medi
  - Ogni file può contenere un solo oggetto codificato in DER
- Privacy Enhanced Mail (PEM)
  - Formato di default usato da OpenSSL
  - Usato per codificare in formato testuale (Base64) oggetti codificati tramite DER
  - Può contenere commenti
  - Ogni file può contenere oggetti multipli codificati in PEM
    - Uno sotto l'altro
  - Sintassi

È possibile convertire un formato nell'altro e viceversa

```
-----BEGIN keyword-----
Base 64 encoding of BER/DER
-----END keyword-----
```

DER vs. PEM

- Distinguished Encoding Rules (DER)
  - Sottoinsieme di BER
  - Codifica in binario i dati descritti medi
  - Ogni file può contenere un solo oggetto codificato in DER
- Privacy Enhanced Mail (PEM)
  - Formato di default usato da OpenSSL
  - Usato per codificare in formato testuale (Base64) oggetti codificati tramite DER
  - Può contenere commenti
  - Ogni file può contenere oggetti multipli codificati in PEM
    - Uno sotto l'altro
  - Sintassi

Entrambe le codifiche sono supportate da OpenSSL

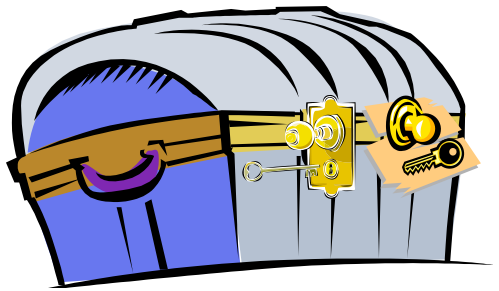
```
-----BEGIN keyword-----
Base 64 encoding of BER/DER
-----END keyword-----
```

PKCS

- Public Key Cryptography Standards (PKCS)
- Insieme di standard sviluppati e pubblicati da RSA Security Inc
- Descrivono, mediante ASN.1, gli oggetti coinvolti in operazioni crittografiche
- Ne esistono vari, ad es.,
  - **PKCS #1** RSA Cryptography Standard
  - **PKCS #3** Diffie-Hellman Key Agreement Standard
  - **PKCS #5** Password-based Encryption Standard
  - **PKCS #7** Cryptographic Message Syntax Standard
  - **PKCS #8** Private-Key Information Syntax Standard
  - **PKCS #12** Personal Information Exchange Syntax Standard

Cifratura RSA in OpenSSL

- OpenSSL fornisce
  - I comandi **genrsa** ed **rsa** per generare, esaminare, manipolare ed usare chiavi RSA
  - Il comando **rsautl** per cifrare mediante RSA



OpenSSL™
Cryptography and SSL/TLS Toolkit

Cifratura RSA in OpenSSL

(Generazione Chiavi)

- La coppia di chiavi RSA può essere generata in OpenSSL mediante il comando **genrsa**

Opzioni principali del comando genrsa

```
openssl genrsa [options] [numbits]
```

- **options**
  - **-des, -des3, -aes128, -aes192, -aes256** Cifra le chiavi generate, utilizzando DES o 3DES, oppure utilizzando AES a 128, 192 o 256 bit
  - **-out file** Scrive in un file le chiavi generate
  - **-passout arg** Password usata per la cifratura delle chiavi
  - **-passin arg** Password usata per la decifratura delle chiavi
  - **-F4,-3** Esponente pubblico: in OpenSSL può essere 65537 oppure 3. Il valore di default è 65537 e non sono ammessi altri valori
- **numbits**
  - Numero di bit del modulo RSA, di default è 512

Cifratura RSA in OpenSSL

(Generazione Chiavi)

- La coppia di chiavi RSA può essere generata in OpenSSL mediante il comando **genrsa**

Opzioni principali del comando genrsa

```
openssl genrsa [options] [numbits]
```

- **options**
  - **-des, -des3, -aes128,** i generate, utilizzando DES o 3DES, o 256 bit
  - **-out file** Scrive in un f
  - **-passout arg** Password
  - **-passin arg** Password usata per la decifratura delle chiavi
  - **-F4,-3** Esponente pubblico: in OpenSSL può essere 65537 oppure 3. Il valore di default è 65537 e non sono ammessi altri valori
- **numbits**
  - Numero di bit del modulo RSA, di default è 512

Per ottenere la lista completa delle opzioni del comando **genrsa** è possibile utilizzare **man genrsa**

Cifratura RSA in OpenSSL

(Generazione Chiavi)

- La coppia di chiavi RSA può essere generata in OpenSSL mediante il comando **genrsa**

Opzioni principali del comando genrsa

```
openssl genrsa [options] [numbits]
```

- **options**
  - **-des, -des3, -aes128, -aes192, -aes256** Cifra le chiavi generate, utilizzando DES o 3DES, oppure utilizzando AES a 128, 192 o 256 bit
  - **-out file** Scrive in un file le chiavi generate
  - **-passout arg** Password usata per la cifratura delle chiavi
  - **-passin arg** Password usata per la decifratura delle chiavi
  - **-F4,-3** Esponente pubblico: in OpenSSL può essere 65537 oppure 3. Il valore di default è 65537 e non sono ammessi altri valori
- **numbits**
  - Numero di bit del modulo RSA, di default è 512

Cifratura RSA in OpenSSL

(Esempio Generazione Chiavi)

Mediante il seguente comando è possibile generare una coppia di chiavi RSA a 1024 bit

- Tale coppia sarà cifrata attraverso AES a 128 bit, usando la stringa "**P1pp0B4ud0**" come password. Il risultato verrà scritto nel file **rsaprivatekey.pem**

```
openssl genrsa -out rsaprivatekey.pem -passout
pass:P1pp0B4ud0 -aes128 1024
```

Se la generazione andrà a buon fine, verrà mostrato il seguente messaggio

```
Generating RSA private key, 1024 bit long modulus
....++++++
.............++++++
e is 65537 (0x10001)
```

Cifratura RSA in OpenSSL

(Esempio Generazione Chiavi)

È possibile visualizzare il contenuto del file **rsaprivatekey.pem** mediante il seguente comando

```
openssl rsa -in rsaprivatekey.pem -text
```

N.B. Per visualizzare il contenuto del file **rsaprivatekey.pem** è necessario conoscere la password mediante la quale è stato cifrato tale file

Le chiavi RSA in OpenSSL sono rappresentate secondo lo standard PKCS #1 (RFC3447)

- https://tools.ietf.org/html/rfc3447

Cifratura RSA in OpenSSL

(Esempio Generazione Chiavi)

È possibile visualizzare il contenuto del file **rsaprivatekey.pem** mediante il seguente comando

```
openssl rsa -in rsaprivatekey.pem -text
```

N.B. Per visualizzare il cor**ivatekey.pem** è necessario conoscere la pa**le** è stato cifrato tale file

Per ottenere la lista completa delle opzioni del comando **rsa** è possibile utilizzare **man rsa**

Le chiavi RSA in OpenSSL sono rappresentate secondo lo standard PKCS #1 (RFC3447)

- https://tools.ietf.org/html/rfc3447

Cifratura RSA in OpenSSL

(Esempio Generazione Chiavi)

Stampa in formato testuale le varie componenti della chiave pubblica o privata, oltre alla versione codificata in formato PEM di tali componenti

È possibile visualizzare **ekey.pem** mediante il seguente co

```
openssl rsa -in rsaprivatekey.pem -text
```

N.B. Per visualizzare il contenuto del file **rsaprivatekey.pem** è necessario conoscere la password mediante la quale è stato cifrato tale file

Le chiavi RSA in OpenSSL sono rappresentate secondo lo standard PKCS #1 (RFC3447)

- https://tools.ietf.org/html/rfc3447

Contenuto del file rsaprivatekey.pem – 1/2

```
Enter pass phrase for rsaprivatekey.pem:
Private-Key: (1024 bit)
modulus:
    00:b6:d3:9a:24:04:87:57:db:88:d4:5e:19:f2:6e:
    31:2d:be:88:3f:78:a4:5c:59:1d:62:aa:fa:a5:ee:
    24:56:f9:9c:57:65:8c:92:66:a7:99:4b:31:0e:45:
    10:65:b3:c4:76:23:e1:66:ca:34:ba:59:6c:d7:38:
    ce:6c:20:e6:d7:59:be:92:c5:0f:87:0e:29:d2:18:
    fb:d4:3e:f9:8c:f1:09:89:c2:b5:fd:d8:6c:64:fa:
    a8:3e:cf:11:60:b3:fa:c0:97:a3:65:41:5a:d1:a5:
    e6:b4:a4:12:e0:11:c6:45:84:23:e9:2f:ee:52:b9:
    f5:53:69:98:82:73:4e:e1:01
publicExponent: 65537 (0x10001)
privateExponent:
    79:15:33:64:df:4a:f3:b9:05:aa:00:98:96:a2:bc:
    17:5b:b4:c8:b4:91:c6:47:8e:da:a5:fa:52:a2:a1:
    96:a8:c2:0e:6f:c6:e5:be:ef:08:eb:6f:95:cf:cf:
    c8:4b:82:b5:eb:95:80:8a:32:d9:a8:18:19:82:19:
    d4:2b:06:36:42:3a:47:1b:e1:80:7d:54:9c:97:f5:
    1f:08:ac:cb:ff:7a:a3:42:51:d5:dd:1e:fa:33:bc:
    50:25:89:3d:42:18:8e:d5:f8:e5:2c:0c:d8:19:98:
    7e:50:94:ae:d4:05:95:ba:9c:9e:87:5f:10:52:50:
    33:14:e8:7b:3e:06:0b:f1
prime1:
    00:eb:50:2b:07:f5:0c:9e:77:3c:35:d9:6c:fe:fd:
    51:24:3a:ff:cf:87:15:17:41:f9:cb:17:00:37:4b:
    1f:d6:5c:e7:ce:a2:13:36:a0:e8:f6:47:d8:78:61:
    4e:39:7d:63:2e:31:94:bd:e5:95:30:36:86:53:33:
    fd:e5:3c:bf:13
prime2:
    00:c6:e6:32:f3:32:6c:19:5d:ec:cc:43:6a:36:44:
    29:9b:d6:54:ee:b1:ba:36:8e:56:07:f6:f3:9b:0b:
    ab:45:57:58:f8:85:89:68:4e:11:52:37:13:fb:d4:
    ad:b4:05:5a:47:1d:e5:ea:64:84:27:4a:a5:e4:e6:
    09:e2:43:9e:1b
exponent1:
    39:9f:e3:39:ca:9f:b1:7a:bf:8a:ec:58:cd:c4:d9:
    07:07:4f:b7:d1:7b:af:bb:5f:61:85:9b:6e:ee:fd:
    b7:ef:fe:70:52:ae:63:d5:59:d7:5d:d6:bb:fc:10:
    3c:f1:e0:c4:e2:2d:6b:a7:7a:36:59:53:e1:b2:3c:
    bd:d3:77:73
exponent2:
    0f:f7:78:cd:97:15:a0:6f:ae:cb:b2:f9:ba:c3:7c:
    07:9b:8c:13:e1:46:8e:8c:9c:91:65:1c:a7:2c:a7:
    a2:18:61:f7:09:59:3f:7b:4c:de:a9:b7:3b:f7:15:
    be:a6:d3:59:74:27:f9:c9:f9:e5:e1:93:31:ad:d9:
    cb:45:1c:53
coefficient:
    56:39:ae:68:3f:99:f7:26:c0:e3:00:3c:95:2f:42:
    f1:b2:f9:14:c7:65:7a:15:f0:04:f2:4b:25:15:18:
    cc:64:27:07:58:47:7e:27:a8:9a:aa:ab:2e:22:57:
    68:e0:47:a4:68:54:23:cb:7d:78:46:01:4e:0f:71:
    60:ea:cb:04
```

Modulo n di 1024 bit

Esponente pubblico e

Esponente privato d

Primo p

Primo q

$d_p = d \bmod (p - 1)$

$d_q = d \bmod (q - 1)$

$q_{inv} = q^{-1} \bmod p$

Chiave privata RSA, espressa secondo la notazione definita da PKCS #1

Contenuto del file rsaprivatekey.pem – 1/2

```
Enter pass phrase for rsaprivatekey.pem:
Private-Key: (1024 bit)
modulus:
    00:b6:d3:9a:24:04:87:57:db:88:d4:5e:19:f2:6e:
    31:2d:be:88:3f:78:a4:5c:59:1d:62:aa:fa:a5:ee:
    24:56:f9:9c:57:65:8c:92:66:a7:99:4b:31:0e:45:
    10:65:b3:c4:76:23:e1:66:ca:34:ba:59:6c:d7:38:
    ce:6c:20:e6:d7:59:be:92:c5:0f:87:0e:29:d2:18:
    fb:d4:3e:f9:8c:f1:09:89:c2:b5:fd:d8:6c:64:fa:
    a8:3e:cf:11:60:b3:fa:c0:97:a3:65:41:5a:d1:a5:
    e6:b4:a4:12:e0:11:c6:45:84:23:e9:2f:ee:52:b9:
    f5:53:69:98:82:73:4e:e1:01
publicExponent: 65537 (0x10001)
privateExponent:
    79:15:33:64:df:4a:f3:b9:05:aa:00:98:96:a2:bc:
    17:5b:b4:c8:b4:91:c6:47:8e:da:a5:fa:52:a2:a1:
    96:a8:c2:0e:6f:c6:e5:be:ef:08:eb:6f:95:cf:cf:
    c8:4b:82:b5:eb:95:80:8a:32:d9:a8:18:19:82:19:
    d4:2b:06:36:42:3a:47:1b:e1:80:7d:54:9c:97:f5:
    1f:08:ac:cb:ff:7a:a3:42:51:d5:dd:1e:fa:33:bc:
    50:25:89:3d:42:18:8e:d5:f8:e5:2c:0c:d8:19:98:
    7e:50:94:ae:d4:05:95:ba:9c:9e:87:5f:10:52:50:
    33:14:e8:7b:3e:06:0b:f1
prime1:
    00:eb:50:2b:07:f5:0c:9e:77:3c:35:d9:6c:fe:fd:
    51:24:3a:ff:cf:87:15:17:41:f9:cb:17:00:37:4b:
    1f:d6:5c:e7:ce:a2:13:36:a0:e8:f6:47:d8:78:61:
    4e:39:7d:63:2e:31:94:bd:e5:95:30:36:86:53:33:
    fd:e5:3c:bf:13
prime2:
    00:c6:e6:32:f3:32:6c:19:5d:ec:cc:43:6a:36:44:
    29:9b:d6:54:ee:b1:ba:36:8e:56:07:f6:f3:9b:0b:
    ab:45:57:58:f8:85:89:68:4e:11:52:37:13:fb:d4:
    ad:b4:05:5a:47:1d:e5:ea:64:84:27:4a:a5:e4:e6:
    09:e2:43:9e:1b
exponent1:
    39:9f:e3:39:ca:9f:b1:7a:bf:8a:ec:58:cd:c4:d9:
    07:07:4f:b7:d1:7b:af:bb:5f:61:85:9b:6e:ee:fd:
    b7:ef:fe:70:52:ae:63:d5:59:d7:5d:d6:bb:fc:10:
    3c:f1:e0:c4:e2:2d:6b:a7:7a:36:59:53:e1:b2:3c:
    bd:d3:77:73
exponent2:
    0f:f7:78:cd:97:15:a0:6f:ae:cb:b2:f9:ba:c3:7c:
    07:9b:8c:13:e1:46:8e:8c:9c:91:65:1c:a7:2c:a7:
    a2:18:61:f7:09:59:3f:7b:4c:de:a9:b7:3b:f7:15:
    be:a6:d3:59:74:27:f9:c9:f9:e5:e1:93:31:ad:d9:
    cb:45:1c:53
coefficient:
    56:39:ae:68:3f:99:f7:26:c0:e3:00:3c:95:2f:42:
    f1:b2:f9:14:c7:65:7a:15:f0:04:f2:4b:25:15:18:
    cc:64:27:07:58:47:7e:27:a8:9a:aa:ab:2e:22:57:
    68:e0:47:a4:68:54:23:cb:7d:78:46:01:4e:0f:71:
    60:ea:cb:04
```

Modulo n di 1024 bit

Esponente pubblico e

Esponente privato d

Primo p

Primo q

$d_p = d \bmod (p - 1)$

$d_q = d \bmod (q - 1)$

$q_{inv} = q^{-1} \bmod p$

Chiave privata RSA, espressa secondo la notazione definita da PKCS #1

Formula di Garner per la decifratura

Calcolo di $c^d \bmod n$

$m_1 = c^{dp} \bmod p$

$m_2 = c^{dq} \bmod q$

$h = q_{inv} \cdot (m_1 - m_2) \bmod p$

$m = m_2 + h \cdot q$

Contenuto del file rsaprivatekey.pem – 2/2

```
writing RSA key
-----BEGIN RSA PRIVATE KEY-----
MIICWwIBAAKBgQC205okBIdX24jUXhnybjEtvog/eKRcWR1iqvql7iRW+ZxXZYyS
ZqeZSzEORRBls8R2I+FmyjS6WWzXOM5sIObXWb6SxQ+HDinSGPvUPvmM8QmJwrX9
2Gxk+qg+zxFgs/rAl6NlQVrRpea0pBLgEcZFhCPpL+5SufVTaZiCc07hAQIDAQAB
AoGAeRUzZN9K87kFqgCYlqK8F1u0yLSRxkeO2qX6UqKhlqjCDm/G5b7vCOtvlc/P
yEuCteuVgIoy2agYGYIZ1CsGNkI6RxvhgH1UnJf1Hwisy/96o0JR1d0e+jO8UCWJ
PUIYjtX45SwM2BmYflCUrtQFlbqcnodfEFJQMxToez4GC/ECQQDrUCsH9Qyedzw1
2Wz+/VEkOv/PhxUXQfnLFwA3Sx/WXOfOohM2oOj2R9h4YU45fWMuMZS95ZUwNoZT
M/3lPL8TAkEAxuYy8zJsGV3szENqNkQpm9ZU7rG6No5WB/bzmwurRVdY+IWJaE4R
UjcT+9SttAVaRx3l6mSEJ0ql5OYJ4kOeGwJAOZ/jOcqfsXq/iuxYzcTZBwdPt9F7
r7tfYYWbbu79t+/+cFKuY9VZ113Wu/wQPPHgxOIta6d6NllT4bI8vdN3cwJAD/d4
zZcVoG+uy7L5usN8B5uME+FGjoyckWUcpyynohhh9wlZP3tM3qm3O/cVvqbTWXQn
+cn55eGTMa3Zy0UcUwJAVjmuaD+Z9ybA4wA8lS9C8bL5FMdlehXwBPJLJRUYzGQn
B1hHfieomqqrLiJXaOBHpGhUI8t9eEYBTg9xYOrLBA==
-----END RSA PRIVATE KEY-----
```

Codifica PEM della chiave privata RSA

Cifratura RSA in OpenSSL

(Esempio Generazione Chiavi)

Mediante il seguente comando è possibile estrarre la chiave pubblica RSA memorizzata nel file **rsaprivatekey.pem**

- Decifrando il contenuto di tale file attraverso la password "**P1pp0B4ud0**" e scrivendo la chiave pubblica nel file **rsapublickey.pem**

```
openssl rsa –in rsaprivatekey.pem –passin pass:P1pp0B4ud0
–pubout –out rsapublickey.pem
```

Per visualizzare il contenuto del file **rsapubkey.pem** è possibile utilizzare il seguente comando

```
openssl rsa –pubin –in rsapubkey.pem –text
```

Cifratura RSA in OpenSSL

(Esempio Generazione Chiavi)

Mediante il seguente comando è possibile estrarre la chiave pubblica RSA memorizzata nel file **rsaprivatekey.pem**

- Decifrando il contenuto di tale file attraverso la password "**P1pp0B4ud0**" e scrivendo la chiave pubblica nel file **rsapublickey.pem**

```
openssl rsa
-pubout -ou
```

N.B. L'opzione -pubin deve essere sempre inclusa nel comando OpenSSL ogniqualvolta il file preso in input sia una chiave pubblica

- Di default OpenSSL si aspetta di ricevere in input una chiave privata

Per visualizzare il contenu sapubkey.pem è possibile utilizzare il seguente comando

```
openssl rsa -pubin -in rsapubkey.pem -text
```

Cifratura RSA in OpenSSL

(Esempio Generazione Chiavi)

Mediante il seguente comando è possibile leggere una coppia di chiavi RSA memorizzate nel file **rsaprivatekey.pem**

➢Decifrando il contenuto di tale file attraverso la password "**P1pp0B4ud0**" e scrivendo la chiave pubblica nel file **rsapublickey.pem**

```
openssl rsa -in rsaprivatekey.pem -passin pass:P1pp0B4ud0
-pubout -out rsapublickey.pem
```

```
unable to load Private Key
139762194448088:error:0906D06C:PEM routines:PEM_read_bio:no
start line:pem_lib.c:701:Expecting: ANY PRIVATE KEY
```

Per visualizzare il contenuto del file **rsapubkey.pem** è possibile utilizzare il seguente comando:

```
openssl rsa -in rsapubkey.pem -text
```

Se si prova a leggere una chiave pubblica RSA non specificando l'opzione -pubin viene restituito un errore

Cifratura RSA in OpenSSL

(Contenuto del file rsapubkey.pem)

```
Modulus (1024 bit):
    00:b6:d3:9a:24:04:87:57:db:88:d4:5e:19:f2:6e:
    31:2d:be:88:3f:78:a4:5c:59:1d:62:aa:fa:a5:ee:
    24:56:f9:9c:57:65:8c:92:66:a7:99:4b:31:0e:45:
    10:65:b3:c4:76:23:e1:66:ca:34:ba:59:6c:d7:38:
    ce:6c:20:e6:d7:59:be:92:c5:0f:87:0e:29:d2:18:
    fb:d4:3e:f9:8c:f1:09:89:c2:b5:fd:d8:6c:64:fa:
    a8:3e:cf:11:60:b3:fa:c0:97:a3:65:41:5a:d1:a5:
    e6:b4:a4:12:e0:11:c6:45:84:23:e9:2f:ee:52:b9:
    f5:53:69:98:82:73:4e:e1:01
Exponent: 65537 (0x10001)
writing RSA key
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC205okBIdX24jUXhnybjEtvog/
eKRcWR1iqvql7iRW+ZxXZYySZqeZSzEORRBls8R2I+FmyjS6WWzXOM5sIObXWb6S
xQ+HDinSGPvUPvmM8QmJwrX92Gxk+qg+zxFgs/rAl6NlQVrRpea0pBLgEcZFhCPp
L+5SufVTaZiCc07hAQIDAQAB
-----END PUBLIC KEY-----
```

Modulo n di 1024 bit

Esponente pubblico e

Chiave pubblica RSA codificata in Base64

Cifratura RSA in OpenSSL

(Il Comando `rsautl`)

- Il comando **`rsautl`** consente di usare le chiavi RSA per la cifratura
  - Permette di specificare opzioni per cifrare e decifrare i dati
- Tale comando non può usato per cifrare grosse moli di dati
  - Per tale scopo va invece usata la cifratura simmetrica (comando **enc**)
- Cifratura e decifratura RSA sono onerose

Cifratura RSA in OpenSSL

Opzioni principali del comando rsautl

```
openssl rsautl [options]
```

- **options**
  - **-in file** File di input
  - **-out file** File di output
  - **-encrypt** Cifra con la chiave pubblica
  - **-decrypt** Decifra con la chiave privata
  - **-sign** Firma con la chiave privata
  - **-verify** Verifica con la chiave pubblica
  - **-inkey** Chiave presa in input
  - **-passin arg** Sorgente da cui deve essere letta la password
  - **-pubin** Specifica che l'input è una chiave pubblica RSA
  - **-pkcs, -oaep, -ssl, -raw** Padding da usare: PKCS#1 v1.5 (default), PKCS#1 OAEP, modalità speciale di padding usata in SSL v2, nessun padding, rispettivamente

Cifratura RSA in OpenSSL

Opzioni principali del comando rsautl

```
openssl rsautl [options]
```

- **options**
  - **-in file** File di input
  - **-out file** File di output
  - **-encrypt** Cifra con la chiave pubblica
  - **-decrypt** Decifra con la chiave privata
  - **-sign** Firma con la chiave privata
  - **-verify** Verifica con la chiave pubblica
  - **-inkey** Chiave presa in input
  - **-passin arg** Sorgente da cui deve essere letta la password
  - **-pubin** Specifica che l'input è una chiave pubblica RSA
  - **-pkcs, -oaep, -ssl, -raw** Padding da usare: PKCS#1 v1.5 (default), PKCS#1 OAEP, modalità speciale di padding usata in SSL v2, nessun padding, rispettivamente

Li vedremo nelle prossime lezioni, quando verrà trattata la firma RSA

Cifratura RSA in OpenSSL

Opzioni principali del comando rsautl

```
openssl rsautl [options]
```

- **options**
  - **-in file** File di input
  - **-out file** File di outpu
  - **-encrypt** Cifra con la c
  - **-decrypt** Decifra con l
  - **-sign** Firma con la chiave privata
  - **-verify** Verifica con la chiave pubblica
  - **-inkey** Chiave presa in input
  - **-passin arg** Sorgente da cui deve essere letta la password
  - **-pubin** Specifica che l'input è una chiave pubblica RSA
  - **-pkcs, -oaep, -ssl, -raw** Padding da usare: PKCS#1 v1.5 (default), PKCS#1 OAEP, modalità speciale di padding usata in SSL v2, nessun padding, rispettivamente

Per ottenere la lista completa delle opzioni del comando **rsautl** è possibile utilizzare **man rsautl**

Cifratura RSA in OpenSSL

(Esempio Cifratura/Decifratura)

testoInChiaro.txt

```
Sicurezza su Reti, A.A. 2016/2017, Prof. Alfredo De Santis.
```

- Usando la chiave pubblica contenuta in **rsapublickey.pem**, il contenuto del file **testoInChiaro.txt** è cifrato e scritto nel file **testoCifrato.txt**

```
openssl rsautl -encrypt -pubin -inkey rsapublickey.pem
-in testoInChiaro.txt -out testoCifrato.txt
```

- Usando la chiave privata contenuta in **rsaprivatekey.pem**, il contenuto del file **testoCifrato.txt** è decifrato e scritto nel file **testoDecifrato.txt**

```
openssl rsautl -decrypt -inkey rsaprivatekey.pem
-in testoCifrato.txt -out testoDecifrato.txt
```

Cifratura RSA in OpenSSL

(Dimensione Dati da Cifrare)

- RSA permette di cifrare dati la cui dimensione è al più pari a quella della chiave
- Nel caso in cui venga utilizzata cifratura con padding, assumendo l'utilizzo di una chiave di X bit, possono essere cifrati
  - (X / 8) – 11 = Y Byte, se la modalità di padding è PKCS#1 v1.5
    - https://www.ietf.org/rfc/rfc2313.txt
  - (X / 8) – 42 = Y Byte, se la modalità di padding è OAEP
    - https://tools.ietf.org/html/rfc3447#section-7.1

Cifratura RSA in OpenSSL

(Dimensione Dati da Cifrare)

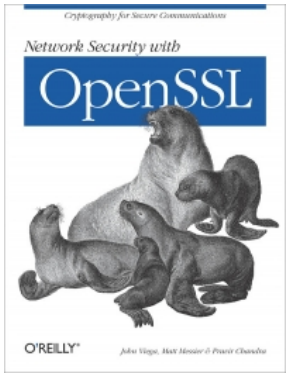
- Nel caso in cui si provi a cifrare un file di dimensioni non conformi rispetto a quelle stabilite dal modulo e dalla modalità di padding, si incorrerà in un errore
- Supponiamo di cifrare un file **bigFile.tar.gz**, di 22952 bit, usando PKCS#1 v1.5 come modalità di padding

```
openssl rsautl -encrypt -pubin -inkey rsapublickey.pem
-in bigFile.tar.gz -out testoCifrato.txt
```

```
RSA operation error
140102711398104:error:0406D06E:rsa
routines:RSA_padding_add_PKCS1_type_2:data too large for key
size:rsa_pk1.c:153:
```

Bibliografia

- Network Security with OpenSSL
  Pravir Chandra, Matt Messier and John Viega (2002), O'Reilly
  - Cap. 2.4.3
  - Appendix A. Command-Line Reference



- Documentazione su OpenSSL
  - https://www.openssl.org/docs/

OpenSSL
Cryptography and SSL/TLS Toolkit

Domande?