# Certificati e PKI in OpenSSL

**Alfredo De Santis**

Dipartimento di Informatica
Università di Salerno

**ads@unisa.it**

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

**Aprile 2017**

# Sommario

- Public Key Infrastructure (PKI) in OpenSSL
  - Creazione della CA
  - Richiesta di Certificato
  - Creazione di un Certificato
  - Verifica di un Certificato
  - Revoca di un Certificato
  - Certificate Revocation List – CRL
- Timestamp in OpenSSL
  - Creazione della TSA
  - Creazione Chiavi e Certificato della TSA
  - Creazione di una Richiesta – Timestamp Request
  - Creazione di una Risposta – Timestamp Response
  - Verifica

# Public Key Infrastructure (PKI)

## (Creazione della CA)

1) Per creare una PKI è innanzitutto necessario stabilire una root directory, in cui risiederanno tutti i file della *Certification Authority (CA)*

```
mkdir CA
cd CA
```

**N.B.** il path verso la directory della CA deve essere anche inserito nel file **openssl.cnf**, mediante il parametro **dir**

```
####################################################################
[ CA_default ]


dir               = .             # Where everything is kept
```

In questo caso, la directory della CA sarà quella corrente, in cui dovrà essere memorizzato il file **openssl.cnf**

OpenSSL Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (Creazione della CA)

2) Nella directory principale della CA andranno create due sottodirectory

- **certs** Utilizzata per conservare una copia di tutti i certificati rilasciati dalla CA
- **private** Utilizzata per mantenere una copia della chiave privata della CA
  - **N.B.** La chiave privata della CA deve essere protetta nel miglior modo possibile
  - Tale chiave dovrebbe essere memorizzata a livello hardware o su una macchina non accessibile dalla rete

```
mkdir certs private
chmod g-rwx,o-rwx private
```

OpenSSL Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (Creazione della *CA*)

3) Nella directory principale della *CA* andranno creati due file, utilizzati dalla *CA* stessa

- **`serial`** Usato per tenere traccia dell'ultimo numero di serie utilizzato per il rilascio di un certificato
  - **N.B.** Due certificati emessi dalla **stessa *CA*** devono **sempre** avere **numeri di serie differenti**
  - Il file **`serial`** sarà inizializzato a contenere il numero 01
- **`certindex.txt`** Sorta di database che tiene traccia dei certificati emessi dalla *CA*
  - OpenSSL richiede che tale file esista (anche se vuoto)

```
echo '01' > serial
touch certindex.txt
```

- Il file **`openssl.cnf`** deve essere copiato nella root directory della *CA*

OpenSSL Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (openssl.cnf)

4) È necessario configurare in maniera opportuna il file **openssl.cnf**

```
dir                                     = .

[ ca ]
default_ca                              = CA_default

[ CA_default ]
serial                                  = $dir/serial
database                                = $dir/certindex.txt
new_certs_dir                           = $dir/certs
certificate                             = $dir/cacert.pem
private_key                             = $dir/private/cakey.pem
default_days                            = 365
default_crl_days                        = 30
default_md                              = sha1
```

Root directory della *CA*

OpenSSL
Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (openssl.cnf)

### Parametri principali del file openssl.cnf

```
dir                             = .

[ ca ]
default_ca                      = CA_default

[ CA_default ]
serial                          = $dir/serial
database                        = $dir/certindex.txt
new_certs_dir                   = $dir/certs
certificate                     = $dir/cacert.pem
private_key                     = $dir/private/cakey.pem
default_days                    = 365
default_crl_days                = 30
default_md                      = sha1
```

File contenente i numeri seriali relativi ai certificati rilasciati dalla CA

OpenSSL Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (openssl.cnf)

### Parametri principali del file openssl.cnf

```
dir                             = .

[ ca ]
default_ca                      = CA_default

[ CA_default ]
serial                          = $dir/serial
database                        = $dir/certindex.txt
new_certs_dir                   = $dir/certs
certificate                     = $dir/cacert.pem
private_key                     = $dir/private/cakey.pem
default_days                    = 365
default_crl_days                = 30
default_md                      = sha1
```

File contenente informazioni su ogni certificato rilasciato dalla CA, quali Common Name (CN), Nazione (C), Organizzazione (O), etc

OpenSSL Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (openssl.cnf)

### Parametri principali del file openssl.cnf

```
dir                             = .

[ ca ]
default_ca                      = CA_default

[ CA_default ]
serial                          = $dir/serial
database                        = $dir/certindex.txt
new_certs_dir                   = $dir/certs
certificate                     = $dir/cacert.pem
private_key                     = $dir/private/cakey.pem
default_days                    = 365
default_crl_days                = 30
default_md                      = sha1
```

Directory in cui verranno memorizzati i certificati emessi dalla CA

OpenSSL Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (openssl.cnf)

### Parametri principali del file openssl.cnf

```
dir                             = .

[ ca ]
default_ca                      = CA_default

[ CA_default ]
serial                          = $dir/serial
database                        = $dir/certindex.txt
new_certs_dir                   = $dir/certs
certificate                     = $dir/cacert.pem
private_key                     = $dir/private/cakey.pem
default_days                    = 365
default_crl_days                = 30
default_md                      = sha1
```

Certificato della CA

OpenSSL
Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (openssl.cnf)

### Parametri principali del file openssl.cnf

```
dir                             = .

[ ca ]
default_ca                      = CA_default

[ CA_default ]
serial                          = $dir/serial
database                        = $dir/certindex.txt
new_certs_dir                   = $dir/certs
certificate                     = $dir/cacert.pem
private_key                     = $dir/private/cakey.pem
default_days                    = 365
default_crl_days                = 30
default_md                      = sha1
```

Chiave privata della CA

OpenSSL Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (openssl.cnf)

### Parametri principali del file openssl.cnf

```
dir                             = .

[ ca ]
default_ca                      = CA_default

[ CA_default ]
serial                          = $dir/serial
database                        = $dir/certindex.t
new_certs_dir                   = $dir/certs
certificate                     = $dir/cacert.pem
private_key                     = $dir/private/ca.pem
default_days                    = 365
default_crl_days                = 30
default_md                      = sha1
```

Durata di ciascun certificato emesso

OpenSSL
Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (openssl.cnf)

### Parametri principali del file openssl.cnf

```
dir                             = .

[ ca ]
default_ca                      = CA_default

[ CA_default ]
serial                          = $dir/serial
database                        = $dir/certindex.txt
new_certs_dir                   = $dir/certs
certificate                     = $dir/cacert.pem
private_key                     = $dir/private/ca
default_days                    = 365
default_crl_days                = 30
default_md                      = sha1
```

Numero di giorni dopo i quali è emessa una nuova CRL

OpenSSL Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (openssl.cnf)

### Parametri principali del file openssl.cnf

```
dir                                     = .

[ ca ]
default_ca                              = CA_default

[ CA_default ]
serial                                  = $dir/serial
database                                = $dir/certindex.
new_certs_dir                           = $dir/certs
certificate                             = $dir/cacert.pem
private_key                             = $dir/private/ca
default_days                            = 365
default_crl_days                        = 30
default_md                              = sha1
```

Algoritmo usato per il calcolo del Message Digest

OpenSSL Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (Creazione della CA)

5) Mediante il seguente comando è possibile creare il certificato (*self-signed root certificate*) della CA

```
openssl req -new -x509 -extensions v3_ca -keyout private/cakey.pem
-out cacert.pem -days 365 -config ./openssl.cnf
```

**N.B.** OpenSSL richiede una password per cifrare la chiave privata

- La sicurezza dell'intera CA si basa sulla sua chiave privata
- Se questa chiave è compromessa, viene compromessa l'integrità della CA
  - Tutti i certificati emessi dalla CA, sia prima che dopo la compromissione, non possono più essere considerati attendibili

OpenSSL Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (Creazione della CA)

5) Mediante il seguente comando è possibile creare il certificato (*self-signed root certificate*) della CA

```
openssl req -new -x509 -extensions v3_ca -keyout private/cakey.pem
-out cacert.pem -days 365 -config ./openssl.cnf
```

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

**N.B.** O[illegible] per cifrare la chiave privata

- La si[illegible] sulla sua chiave privata
- Se questa chiave è compromessa, viene compromessa l'integrità della CA
  - Tutti i certificati emessi dalla CA, sia prima che dopo la compromissione, non possono più essere considerati attendibili

OpenSSL Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (Creazione della CA)

Mediante il seguente comando è possibile visualizzare la coppia di chiavi della CA

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

Mediante il seguente comando è possibile visualizzare il certificato della CA

```
openssl x509 -in cacert.pem -text -noout
```

OpenSSL
Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (Creazione della CA)

Mediante il seguente comando [illegible]pia di chiavi della CA

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

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

Mediante il seguente coma[illegible] certificato della CA

Per ottenere la lista completa delle opzioni del comando **x509** è possibile utilizzare **man x509**

```
openssl x509 -in cacert.pem -text -noout
```

OpenSSL Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (cacert.pem)

Certificati e CRL utilizzati in OpenSSL sono conformi allo standard X.509 v3, specificato nell'RFC 5280 e noto come PKIX for Public Key Infrastructure (X.509)

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            9c:1a:a4:06:9b:ae:22:e1
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: O=University of Salerno, L=Fisciano, ST=Italy, C=IT
        Validity
            Not Before: Feb 19 19:55:34 2017 GMT
            Not After : Feb 19 19:55:34 2018 GMT
        Subject: O=University of Salerno, L=Fisciano, ST=Italy, C=IT
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:d1:6b:d7:34:80:7e:8f:94:c7:fd:eb:d1:65:6e:
                    cd:c4:cc:c0:d8:bb:ea:de:32:ee:2a:60:d9:33:bd:
                    c2:61:cc:83:22:9c:88:4f:b0:a8:e0:e6:11:9f:ff:
                    e3:7c:7a:ee:5d:f2:53:ce:9a:9d:e5:e3:5d:35:bf:
                    09:c0:2f:6e:2b:f1:d9:39:2a:85:25:80:0b:2d:56:
                    9b:2d:ed:fc:3c:b6:4c:af:e4:34:9c:95:6e:1f:95:
                    87:20:ba:7f:9e:44:53:43:ea:c4:a3:10:a1:d8:ee:
                    c4:b2:d1:8b:eb:66:93:68:12:ff:6d:7c:04:d3:4d:
                    50:2d:14:cb:b8:7c:69:c9:8b
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:TRUE
            X509v3 Subject Key Identifier:
                53:DD:9D:34:73:41:27:05:9A:54:A1:46:CD:33:03:33:8E:B1:2F:B9
            X509v3 Authority Key Identifier:
                keyid:53:DD:9D:34:73:41:27:05:9A:54:A1:46:CD:33:03:33:8E:B1:
2F:B9
                DirName:/O=University of Salerno/L=Fisciano/ST=Italy/C=IT
                serial:9C:1A:A4:06:9B:AE:22:E1

    Signature Algorithm: sha1WithRSAEncryption
        2b:a1:9f:df:a9:7a:0b:71:11:19:57:5c:5c:98:fd:64:a8:5d:
        3f:8e:ce:37:ff:3b:8c:35:e9:58:02:8f:b0:6c:db:a1:ff:db:
        c7:65:10:14:5b:88:22:ae:89:75:df:0e:4b:df:73:c8:bf:22:
        bc:36:35:96:23:0f:eb:64:74:51:5e:e7:3d:77:6d:3a:53:58:
        71:df:fc:35:28:29:ff:f2:00:ad:fa:a9:db:b6:33:05:5f:f7:
        59:33:2d:aa:28:a5:59:ab:ea:68:ae:a8:b7:3a:6a:33:fd:4c:
        ac:96:86:13:52:9a:1c:0c:8d:cb:06:c4:90:b2:42:ef:67:9f:
        73:24
```

OpenSSL Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (cacert.pem)

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            9c:1a:a4:06:9b:ae:22:e1
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: O=University of Salerno, L=Fisciano, ST=Italy, C=IT
        Validity
            Not Before: Feb 19 1[illegible]:55:34 2017 GMT
            Not After : Feb 19 19:[illegible]:34 2018 GMT
        Subject: O=University of Sale[illegible]L=Fisciano, ST=Italy, C=IT
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryp[illegible]
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:d1:6b:d7:34:80:7e:8f:94:c7:fd[illegible]:6e:
                    cd:c4:cc:c0:d8:bb:ea:de:32[illegible]
                    c2:61:cc:83:22:9c:88:4f:b0[illegible]
                    e3:7c:7a:ee:5d:f2:53:ce:9a[illegible]
                    09:c0:2f:6e:2b:f1:d9:39:2a[illegible]
                    9b:2d:ed:fc:3c:b6:4c:af:e4[illegible]
                    87:20:ba:7f:9e:44:53:43:ea:c4:a3:10:a1:d8:ee:
                    c4:b2:d1:8b:eb:66:93:68:12:ff:6d:7c:04:d3:4d:
                    50:2d:14:cb:b8:7c:69:c9:8b
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:TRUE
            X509v3 Subject Key Identifier:
                53:DD:9D:34:73:41:27:05:9A:54:A1:46:CD:33:03:33:8E:B1:2F:B9
            X509v3 Authority Key Identifier:
                keyid:53:DD:9D:34:73:41:27:05:9A:54:A1:46:CD:33:03:33:8E:B1:
2F:B9
                DirName:/O=University of Salerno/L=Fisciano/ST=Italy/C=IT
                serial:9C:1A:A4:06:9B:AE:22:E1

    Signature Algorithm: sha1WithRSAEncryption
        2b:a1:9f:df:a9:7a:0b:71:11:19:57:5c:5c:98:fd:64:a8:5d:
        3f:8e:ce:37:ff:3b:8c:35:e9:58:02:8f:b0:6c:db:a1:ff:db:
        c7:65:10:14:5b:88:22:ae:89:75:df:0e:4b:df:73:c8:bf:22:
        bc:36:35:96:23:0f:eb:64:74:51:5e:e7:3d:77:6d:3a:53:58:
        71:df:fc:35:28:29:ff:f2:00:ad:fa:a9:db:b6:33:05:5f:f7:
        59:33:2d:aa:28:a5:59:ab:ea:68:ae:a8:b7:3a:6a:33:fd:4c:
        ac:96:86:13:52:9a:1c:0c:8d:cb:06:c4:90:b2:42:ef:67:9f:
        73:24
```

CA che ha emesso il certificato (Issuer)

OpenSSL Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (cacert.pem)

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            9c:1a:a4:06:9b:ae:22:e1
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: O=University of Salerno, L=Fisciano, ST=Italy, C=IT
        Validity
            Not Before: Feb 19 19:55:34 2017 GMT
            Not After : Feb 19 19:55:34 2018 GMT
        Subject: O=University of Salerno, L=Fisciano, ST=Italy, C=IT
        Subject Public Key In
            Public Key Algori      rsaEncryption
            RSA Public Key:      bit)
                Modulus (102
                    00:d1:6b          :7e:8f:94:c7:fd:eb:d1:65:6e:
                                                              bd:
                                                              ff:
                                                              bf:
                                                              56:
                                                              95:
                    87:20:ba:7f:9e:44:53:43:ea:c4:a3:10:a1:d8:ee:
                    c4:b2:d1:8b:eb:66:93:68:12:ff:6d:7c:04:d3:4d:
                    50:2d:14:cb:b8:7c:69:c9:8b
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:TRUE
            X509v3 Subject Key Identifier:
                53:DD:9D:34:73:41:27:05:9A:54:A1:46:CD:33:03:33:8E:B1:2F:B9
            X509v3 Authority Key Identifier:
                keyid:53:DD:9D:34:73:41:27:05:9A:54:A1:46:CD:33:03:33:8E:B1:
2F:B9
                DirName:/O=University of Salerno/L=Fisciano/ST=Italy/C=IT
                serial:9C:1A:A4:06:9B:AE:22:E1

    Signature Algorithm: sha1WithRSAEncryption
        2b:a1:9f:df:a9:7a:0b:71:11:19:57:5c:5c:98:fd:64:a8:5d:
        3f:8e:ce:37:ff:3b:8c:35:e9:58:02:8f:b0:6c:db:a1:ff:db:
        c7:65:10:14:5b:88:22:ae:89:75:df:0e:4b:df:73:c8:bf:22:
        bc:36:35:96:23:0f:eb:64:74:51:5e:e7:3d:77:6d:3a:53:58:
        71:df:fc:35:28:29:ff:f2:00:ad:fa:a9:db:b6:33:05:5f:f7:
        59:33:2d:aa:28:a5:59:ab:ea:68:ae:a8:b7:3a:6a:33:fd:4c:
        ac:96:86:13:52:9a:1c:0c:8d:cb:06:c4:90:b2:42:ef:67:9f:
        73:24
```

Periodo di validità del certificato


# Public Key Infrastructure (PKI)

## (cacert.pem)

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            9c:1a:a4:06:9b:ae:22:e1
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: O=University of Salerno, L=Fisciano, ST=Italy, C=IT
        Validity
            Not Before: Feb 19 19:55:34 2017 GMT
            Not After : Feb 19 19:55:34 2018 GMT
        Subject: O=University of Salerno, L=Fisciano, ST=Italy, C=IT
        Subject Public Key Info:
            Public Key Algorithm: rs                yption
            RSA Public Key: (1024 bi
                Modulus (1024 bit):
                    00:d1:6b:d7:34:80                    :eb:d1:65:6e:
                    cd:c4:
                    c2:61:
                    e3:7c:
                    09:c0:
                    9b:2d:ed:fc:3c:b6:4c:af:e4:34:9c:95:6e:1f:95:
                    87:20:ba:7f:9e:44:53:43:ea:c4:a3:10:a1:d8:ee:
                    c4:b2:d1:8b:eb:66:93:68:12:ff:6d:7c:04:d3:4d:
                    50:2d:14:cb:b8:7c:69:c9:8b
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:TRUE
            X509v3 Subject Key Identifier:
                53:DD:9D:34:73:41:27:05:9A:54:A1:46:CD:33:03:33:8E:B1:2F:B9
            X509v3 Authority Key Identifier:
                keyid:53:DD:9D:34:73:41:27:05:9A:54:A1:46:CD:33:03:33:8E:B1:
2F:B9
                DirName:/O=University of Salerno/L=Fisciano/ST=Italy/C=IT
                serial:9C:1A:A4:06:9B:AE:22:E1

    Signature Algorithm: sha1WithRSAEncryption
        2b:a1:9f:df:a9:7a:0b:71:11:19:57:5c:5c:98:fd:64:a8:5d:
        3f:8e:ce:37:ff:3b:8c:35:e9:58:02:8f:b0:6c:db:a1:ff:db:
        c7:65:10:14:5b:88:22:ae:89:75:df:0e:4b:df:73:c8:bf:22:
        bc:36:35:96:23:0f:eb:64:74:51:5e:e7:3d:77:6d:3a:53:58:
        71:df:fc:35:28:29:ff:f2:00:ad:fa:a9:db:b6:33:05:5f:f7:
        59:33:2d:aa:28:a5:59:ab:ea:68:ae:a8:b7:3a:6a:33:fd:4c:
        ac:96:86:13:52:9a:1c:0c:8d:cb:06:c4:90:b2:42:ef:67:9f:
        73:24
```

Entità associata alla chiave pubblica

OpenSSL
Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (cacert.pem)

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            9c:1a:a4:06:9b:ae:22:e1
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: O=University of Salerno, L=Fisciano, ST=Italy, C=IT
        Validity
            Not Before: Feb 19 19:55:34 2017 GMT
            Not After : Feb 19 19:55:34 2018 GMT
        Subject: O=University of Salerno, L=Fisciano, ST=Italy, C=IT
        Subject Public Key Info:
            Public Key Algorithm: rsaEn          tion
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:d1:6b:d7:34:80:7e                  d1:65:6e:
                    cd:c4:cc:c0:d8:bb:ea:                 3:bd:
                    c2:61:
                    e3:7c:
                    09:c0:
                    9b:2d:
                    87:20:
                    c4:b2:
                    50:2d:
                Exponent:
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:TRUE
            X509v3 Subject Key Identifier:
                53:DD:9D:34:73:41:27:05:9A:54:A1:46:CD:33:03:33:8E:B1:2F:B9
            X509v3 Authority Key Identifier:
                keyid:53:DD:9D:34:73:41:27:05:9A:54:A1:46:CD:33:03:33:8E:B1:
2F:B9
                DirName:/O=University of Salerno/L=Fisciano/ST=Italy/C=IT
                serial:9C:1A:A4:06:9B:AE:22:E1

    Signature Algorithm: sha1WithRSAEncryption
        2b:a1:9f:df:a9:7a:0b:71:11:19:57:5c:5c:98:fd:64:a8:5d:
        3f:8e:ce:37:ff:3b:8c:35:e9:58:02:8f:b0:6c:db:a1:ff:db:
        c7:65:10:14:5b:88:22:ae:89:75:df:0e:4b:df:73:c8:bf:22:
        bc:36:35:96:23:0f:eb:64:74:51:5e:e7:3d:77:6d:3a:53:58:
        71:df:fc:35:28:29:ff:f2:00:ad:fa:a9:db:b6:33:05:5f:f7:
        59:33:2d:aa:28:a5:59:ab:ea:68:ae:a8:b7:3a:6a:33:fd:4c:
        ac:96:86:13:52:9a:1c:0c:8d:cb:06:c4:90:b2:42:ef:67:9f:
        73:24
```

Entità associata alla chiave pubblica

- Per quanto riguarda i certificati *root self-signed*, l'Issuer coincide col Subject

OpenSSL
Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (cacert.pem)

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            9c:1a:a4:06:9b:ae:22:e1
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: O=University of Salerno, L=Fisciano, ST=Italy, C=IT
        Validity
            Not Before: Feb 19 19:55:34 2017 GMT
            Not After : Feb 19 19:55:34 2018 GMT
        Subject: O=University of Salerno, L=Fisciano, ST=Italy, C=IT
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:d1:6b:d7:34:80:7e:8f:94:c7:fd:eb:d1:65:6e:
                    cd:c4:cc:c0:d8:bb:ea:de:32:ee:2a:60:d9:33:bd:
                    c2:61:cc:83:22:9c:88:4f:b0:a8:e0:e6:11:9f:ff:
                    e3:7c:7a:ee:5d:f2:53:ce:9a:9d:e5:e3:5d:35:bf:
                    09:c0:2f:6e:2b:f1:d9:39:2a:85:25:80:0b:2d:56:
                    9b:2d:ed:fc:3c:b6:4c:af:e4:34:9c:95:6e:1f:95:
                    87:20:ba:7f:9e:44:53:43:ea:c4:a3:10:a1:d8:ee:
                    c4:b2:d1:8b:eb:66:93:68:12:ff:6d:7c:04:d3:4d:
                    50:2d:14:cb:b8:7c:69:c9:8b
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:TRUE
            X509v3 Subject Key Identifier:
                53:DD:9D:34:73:41:27:05:9A:54:A1:46:CD                33:8E:B1:2F:B9
            X509v3 Authority Key Identifier:
                keyid:53:DD:9D:34:73:41:27:05:9A:54:A1:46:C                8E:B1:
2F:B9

                DirName:/O=University of Salerno/L=Fisc
                serial:9C:1A:A4:06:9B:AE:22:E1

    Signature Algorithm: sha1WithRSAEncryption
        2b:a1:9f:df:a9:7a:0b:71:11:19:57:5c:5c:98:fd:64:a8:5d:
        3f:8e:ce:37:ff:3b:8c:35:e9:58:02:8f:b0:6c:db:a1:ff:db:
        c7:65:10:14:5b:88:22:ae:89:75:df:0e:4b:df:73:c8:bf:22:
        bc:36:35:96:23:0f:eb:64:74:51:5e:e7:3d:77:6d:3a:53:58:
        71:df:fc:35:28:29:ff:f2:00:ad:fa:a9:db:b6:33:05:5f:f7:
        59:33:2d:aa:28:a5:59:ab:ea:68:ae:a8:b7:3a:6a:33:fd:4c:
        ac:96:86:13:52:9a:1c:0c:8d:cb:06:c4:90:b2:42:ef:67:9f:
        73:24
```

Chiave pubblica del Subject

OpenSSL
Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (cacert.pem)

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            9c:1a:a4:06:9b:ae:22:e1
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: O=University of Salerno, L=Fisciano, ST=Italy, C=IT
        Validity
            Not Before: Feb 19 19:55:34 2017 GMT
            Not After : Feb 19 19:55:34 2018 GMT
        Subject: O=University of Salerno, L=Fisciano, ST=Italy, C=IT
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:d1:6b:d7:34:80:7e:8f:94:c7:fd:eb:d1:65:6e:
                    cd:c4:cc:c0:d8:bb:ea:de:32:ee:2a:60:d9:33:bd:
                    c2:61:cc:83:22:9c:88:4f:b0:a8:e0:e6:11:9f:ff:
                    e3:7c:7a:ee:5d:f2:53:ce:9a:9d:e5:e3:5d:35:bf:
                    09:c0:2f:6e:2b:f1:d9:39:2a:85:25:80:0b:2d:56:
                    9b:2d:ed:fc:3c:b6:4c:af:e4:34:9c:95:6e:1f:95:
                    87:20:ba:
                    c4:b2:d1:
                    50:2d:14:
                Exponent: 655
        X509v3 extensions:
            X509v3 Basic Const
                CA:TRUE
            X509v3 Subject Key
                53:DD:9D:34:73
            X509v3 Authority Key Identi
                keyid:53:DD:9D:34:73:                A:54:A1:46:CD:33:03:33:8E:B1:
2F:B9

                DirName:/O=Univer                 of Salerno/L=Fisciano/ST=Italy/C=IT
                serial:9C:1A:A4                 9B:AE:22:E1

    Signature Algorithm: sha1WithRSAEncryption
        2b:a1:9f:df:a9:7a:0b:71:11:19:57:5c:5c:98:fd:64:a8:5d:
        3f:8e:ce:37:ff:3b:8c:35:e9:58:02:8f:b0:6c:db:a1:ff:db:
        c7:65:10:14:5b:88:22:ae:89:75:df:0e:4b:df:73:c8:bf:22:
        bc:36:35:96:23:0f:eb:64:74:51:5e:e7:3d:77:6d:3a:53:58:
        71:df:fc:35:28:29:ff:f2:00:ad:fa:a9:db:b6:33:05:5f:f7:
        59:33:2d:aa:28:a5:59:ab:ea:68:ae:a8:b7:3a:6a:33:fd:4c:
        ac:96:86:13:52:9a:1c:0c:8d:cb:06:c4:90:b2:42:ef:67:9f:
        73:24
```

Il certificato è firmato con la chiave privata della CA e la firma è mostrata alla fine di tale certificato

OpenSSL
Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (Richiesta di Certificato)

È innanzitutto necessario creare la directory in cui risiederà la chiave privata dell'utente

```
mkdir private
chmod g-rwx,o-rwx private
```

Mediante il seguente comando, un utente, che chiameremo Utente 1, può effettuare una richiesta di certificato

```
openssl req -new -out utente1-req.pem -keyout private/utente1-key.pem
-days 365 -config ./openssl.cnf
```

- Questo comando porterà alla creazione di due file
  - **utente1-req.pem** Contenente la richiesta di certificato
  - **utente1-key.pem** Contenente le chiavi associate alla richiesta di certificato
- N.B. Verrà richiesta una password, che sarà usata per cifrare la chiave privata

OpenSSL Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (Richiesta di Certificato)

Mediante il seguente comando è possibile controllare il contenuto di una richiesta di certificato

```
openssl req -text -noout -in utente1-req.pem
```

```
Certificate Request:
    Data:
        Version: 0 (0x0)
        Subject: O=My Company, L=My Town, ST=State or Providence, C=US
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:c7:95:15:f7:e0:10:94:49:ed:11:6e:2e:10:8b:
                    19:de:47:85:66:d6:a9:e4:d4:a5:a9:19:aa:44:63:
                    6e:ca:21:5a:f5:78:c4:7a:98:2d:2e:e8:f6:71:d1:
                    6d:33:06:6d:74:bf:02:39:93:3c:42:24:85:d8:15:
                    8e:9c:be:65:cd:30:60:97:38:c0:a2:95:67:f0:07:
                    92:e6:bd:7b:0e:a0:7c:a4:a3:8c:ea:eb:8c:d0:18:
                    4a:61:0f:38:d9:19:89:7b:1e:80:53:33:89:18:90:
                    81:ad:f0:dd:b5:03:e2:30:09:6f:94:da:05:a9:c0:
                    8d:f0:0d:22:4a:61:0e:3c:b7
                Exponent: 65537 (0x10001)
        Attributes:
        Requested Extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Subject Key Identifier:
                E0:CE:3E:E5:E4:4C:5F:F1:F6:DC:41:04:C2:8A:57:37:7E:23:FF:D6
    Signature Algorithm: md5WithRSAEncryption
        29:fb:65:36:73:4b:10:10:1c:e0:f9:20:29:6c:82:c5:2d:78:
        2b:15:18:12:f7:99:45:2d:17:5d:61:1c:d6:6d:17:89:8b:ba:
        5b:50:b7:3d:74:ec:c7:1a:01:0d:17:e7:2a:f3:a7:e7:64:b7:
        ce:5c:75:7b:40:df:ec:98:0d:e5:5d:bb:b6:9f:1c:3c:1d:e8:
        72:74:02:f2:ae:30:03:d1:86:b0:1e:1a:2a:86:03:c3:80:ff:
        1b:18:7d:c3:0b:f3:b5:39:4e:6f:34:be:c1:5d:3a:50:7a:02:
        60:99:65:14:8c:87:94:65:88:f5:cb:dd:a2:30:b8:d3:83:66:
        a0:96
```

# Public Key Infrastructure (PKI)

## (Creazione di un Certificato)

OpenSSL fornisce il comando **ca** per implementare le principali mansioni svolte da una *CA*

### Struttura generale del comando ca

```
openssl ca args
```

- **args**
  - **-in file** File di input, contenente la richiesta di certificato
  - **-out file** File di output
  - **-config file** File di configurazione
  - **-gencrl** Genera una nuova CRL
  - **-revoke file** Revoca un certificato passato come file
  - **-updatedb** Rimuove dal database dei certificati quelli scaduti

OpenSSL Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (Creazione di un Certificato)

OpenSSL fornisce il comando **ca** per implementare le principali mansioni svolte da una *CA*

### Struttura generale del comando ca

```
openssl ca args
```

- **args**
  - **-in file** File sta di certificato
  - **-out file** Fil
  - **-config file** File di configurazione
  - **-gencrl** Genera una nuova CRL
  - **-revoke file** Revoca un certificato passato come file
  - **-updatedb** Rimuove dal database dei certificati quelli scaduti

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

OpenSSL Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (Creazione di un Certificato)

L'Utente 1 dovrà inviare alla *CA* il file contenente la richiesta di certificato (**utente1-req.pem**)

La *CA*, non appena ricevuta la richiesta di certificato da parte dell'Utente 1, potrà generare il certificato per tale utente, mediante il seguente comando

```
openssl ca -out utente1-cert.pem -config ./openssl.cnf
-in utente1-req.pem
```

# Public Key Infrastructure (PKI)

## (Creazione di un Certificato)

```
openssl ca -out utente1-cert.pem -config ./openssl.cnf
-in utente1-req.pem
```

**Se il comando va a buon fine**

- Viene aggiunta una nuova entry nel file **certindex.txt**
- Viene incrementato il numero di serie nel file **serial**
- Vengono generati due nuovi file
  - **utente1-req.pem** Certificato per l'Utente 1, che dovrà essere inviato a tale utente
  - Copia di tale certificato, che sarà automaticamente memorizzata nella directory **certs** della CA
    - Il nome di tale file sarà dato dal suo numero seriale, seguito dall'estensione .pem

# Public Key Infrastructure (PKI)

## (Verifica di un Certificato)

Mediante il seguente comando è possibile verificare la validità di un certificato rispetto alla CA che lo ha emesso

```
openssl verify -CAfile cacert.pem utente1-cert.pem
```

```
$ openssl verify -CAfile cacert.pem utente1-cert.pem
utente1-cert.pem: OK
$
```

# Public Key Infrastructure (PKI)

## (PKCS #12)

**Osservazione**: L'Utente 1 può esportare il proprio certificato in formato PKCS #12

- Questo formato permette di combinare in un unico file
  - La chiave pubblica dell'Utente 1
  - La sua chiave privata
  - Il certificato della *CA*

```
openssl pkcs12 -export -in utente1-cert.pem -inkey
private/utente1-key.pem -certfile cacert.pem -name
"Certificato Utente1" -out utente1-cert.p12
```

OpenSSL Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (Revoca di un Certificato)

- La revoca di un certificato è un processo semplice, che richiede solo una copia del certificato da revocare
  - La CA possiede una copia di tutti i certificati che ha emesso
- Mediante il seguente comando la CA può revocare un certificato emesso

```
openssl ca -revoke utente1-cert.pem -config ./openssl.cnf
```

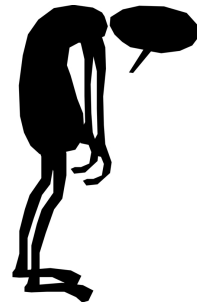
OpenSSL Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (Revoca di un Certificato)

- **Osservazioni**
  - Su un certificato revocato non avviene alcuna modifica
  - L'unico cambiamento evidente è nel database della *CA* (**`certindex.txt`**)
    - L'entry corrispondente al certificato revocato inizia con la lettera **R**, mentre quella corrispondente ad un certificato valido inizia con la lettera **V**
  - Una volta che un certificato è stato rilasciato, non può essere più modificato
    - Non è possibile aggiornare tutte le copie di un dato certificato emesso!



# Public Key Infrastructure (PKI)

## (Certificate Revocation List - CRL)

- Il problema della revoca dei certificati può essere risolto mediante l'utilizzo di una *Certificate Revocation List (CRL)*
  - Senza l'utilizzo di una CRL, l'unica entità a conoscenza della revoca di un determinato certificato è la CA coinvolta nella revoca

# Public Key Infrastructure (PKI)

## (Certificate Revocation List - CRL)

- Mediante il seguente comando è possibile generare la CRL

```
openssl ca -gencrl -out expampleca.crl -config ./openssl.cnf
```

  - Se il comando è eseguito senza errori, non viene mostrato alcun output, altrimenti, viene mostrato un messaggio d'errore

- Mediante il seguente comando è possibile visualizzare il contenuto della CRL

```
openssl crl -in expampleca.crl -text
```

OpenSSL Cryptography and SSL/TLS Toolkit

PKI

# Public Key Infrastructure (PKI)

## (Certificate Revocation List - CRL)

- Mediante il seguente comando è possibile generare la CRL

```
openssl ca -gencrl -out expampleca.crl -config ./openssl.cnf
```

  - Se il comando è eseguito senza errori, non viene mostrato alcun output, altrimenti, viene mostrato un messaggio d'errore

- Mediante il [illegible] possibile visualizzare il co[illegible]enuto della CRL

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

```
openssl crl -in expampleca.crl -text
```

OpenSSL Cryptography and SSL/TLS Toolkit

PKI

# exampleca.crl

```
Certificate Revocation List (CRL):
        Version 1 (0x0)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: /O=University of Salerno/L=Fisciano/ST=Italy/C=IT
        Last Update: Feb 20 21:17:00 2017 GMT
        Next Update: Mar 22 21:17:00 2017 GMT
Revoked Certificates:
    Serial Number: 01
        Revocation Date: Feb 20 20:33:51 2017 GMT
    Serial Number: 02
        Revocation Date: Feb 20 21:11:00 2017 GMT
    Serial Number: 03
        Revocation Date: Feb 20 21:15:26 2017 GMT
    Signature Algorithm: sha1WithRSAEncryption
        66:77:28:17:7f:d5:38:1c:5a:e9:74:cc:e0:6d:e0:78:79:5a:
        62:e0:fc:53:29:d9:0a:ea:9e:85:df:69:f4:e8:03:76:12:d6:
        96:1c:c4:fb:68:af:e4:b4:12:df:96:bd:ee:d8:68:ae:4b:81:
        30:51:fb:16:5b:42:fc:55:c1:44:ca:1a:e2:b8:4a:cf:41:51:
        2f:3d:72:48:14:9a:1f:a6:8d:65:97:86:81:3c:05:a2:62:11:
        08:0f:61:17:be:63:2a:26:24:18:75:79:7e:7d:7d:ec:5e:21:
        03:e2:b6:38:b8:6b:ec:c1:7b:bd:42:99:f7:a9:48:4f:94:9d:
        5f:17
-----BEGIN X509 CRL-----
MIIBUzCBvTANBgkqhkiG9w0BAQUFADBQMR4wHAYDVQQKExVVbml2ZXJzaXR5IG9m
IFNhbGVybm8xETAPBgNVBAcTCEZpc2NpYW5vMQ4wDAYDVQQIEwVJdGFseTELMAkG
A1UEBhMCSVQXDTE3MDIyMDIxMTcwMFoXDTE3MDMyMjIxMTcwMFowPDASAgEBFw0x
NzAyMjAyMDMzNTFaMBICAQIXDTE3MDIyMDIxMTEwMFowEgIBAxcNMTcwMjIwMjEx
NTI2WjANBgkqhkiG9w0BAQUFAAOBgQBmdygXf9U4HFrpdMzgbeB4eVpi4PxTKdkK
6p6F32n06AN2EtaWHMT7aK/ktBLflr3u2GiuS4EwUfsWW0L8VcFEyhriuErPQVEv
PXJIFJofpo1ll4aBPAWiYhEID2EXvmMqJiQYdXl+fX3sXiED4rY4uGvswXu9Qpn3
qUhPlJ1fFw==
-----END X509 CRL-----
```

OpenSSL
Cryptography and SSL/TLS Toolkit

PKI

# exampleca.crl

```
Certificate Revocation List (CRL):
        Version 1 (0x0)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: /O=University of Salerno/L=Fisciano/ST=Italy/C=IT
        Last Update: Feb 20 21:17:00 2017 GMT
        Next Update: Mar 22 21:17:00 2017 GMT
Revoked Certificates:
    Serial Number: 01
        Revocation Date: Feb 20 20:33:51 2017 GMT
    Serial Number: 02
        Revocation Date: Feb 20 21:11:00 2017 GMT
    Serial Number: 03
        Revocation Date: Feb 20 21:15:26 2017 GMT
    Signature Algorithm: sha1WithRSAEncryption
        66:77:28:17:7f:d5:38:1c:[illegible]:e9:74:cc:e0:6d:e0:78:79:5a:
        62:e0:fc:53:29:d9:0a:ea:9e[illegible]df:69:f4:e8:03:76:12:d6:
        96:1c:c4:fb:68:af:e4:b4:12:d[illegible]:ee:d8:68:ae:4b:81:
        30:51:fb:16:5b:42:fc:55:c1[illegible]:
        2f:3d:72:48:14:9a:1f:a6:8c[illegible]:
        08:0f:61:17:be:63:2a:26:24[illegible]:
        03:e2:b6:38:b8:6b:ec:c1:7b[illegible]:
        5f:17
-----BEGIN X509 CRL-----
MIIBUzCBvTANBgkqhkiG9w0BAQUFADBQMR4wHAYDVQQKExVVbml2ZXJzaXR5IG9m
IFNhbGVybm8xETAPBgNVBAcTCEZpc2NpYW5vMQ4wDAYDVQQIEwVJdGFseTELMAkG
A1UEBhMCSVQXDTE3MDIyMDIxMTcwMFoXDTE3MDMyMjIxMTcwMFowPDASAgEBFw0x
NzAyMjAyMDMzNTFaMBICAQIXDTE3MDIyMDIxMTEwMFowEgIBAxcNMTcwMjIwMjEx
NTI2WjANBgkqhkiG9w0BAQUFAAOBgQBmdygXf9U4HFrpdMzgbeB4eVpi4PxTKdkK
6p6F32n06AN2EtaWHMT7aK/ktBLflr3u2GiuS4EwUfsWW0L8VcFEyhriuErPQVEv
PXJIFJofpo1ll4aBPAWiYhEID2EXvmMqJiQYdXl+fX3sXiED4rY4uGvswXu9Qpn3
qUhPlJ1fFw==
-----END X509 CRL-----
```

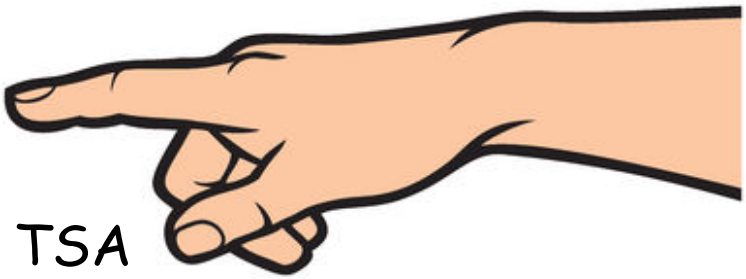
Numero seriale dei certificati revocati e relativa data di revoca

OpenSSL Cryptography and SSL/TLS Toolkit

PKI

# Sommario

- Public Key Infrastructure (PKI) in OpenSSL
  - Creazione della CA
  - Richiesta di Certificato
  - Creazione di un Certificato
  - Verifica di un Certificato
  - Revoca di un Certificato
  - Certificate Revocation List – CRL
- Timestamp in OpenSSL
  - Creazione della TSA
  - Creazione Chiavi e Certificato della TSA
  - Creazione di una Richiesta – Timestamp Request
  - Creazione di una Risposta – Timestamp Response
  - Verifica



# Timestamp in OpenSSL
## (RFC 3161)

- Una *Time Stamping Authority (TSA)*
  - Può essere parte di una PKI
  - Certifica l'esistenza di un certo dato, prima di uno specifico istante temporale
- Il comando **ts** permette di realizzare una TSA in OpenSSL
  - La TSA fornita da OpenSSL implementa le specifiche dell'*RFC 3161 (Time-Stamp Protocol, TSP)* ed è basata sul modello Client/Server

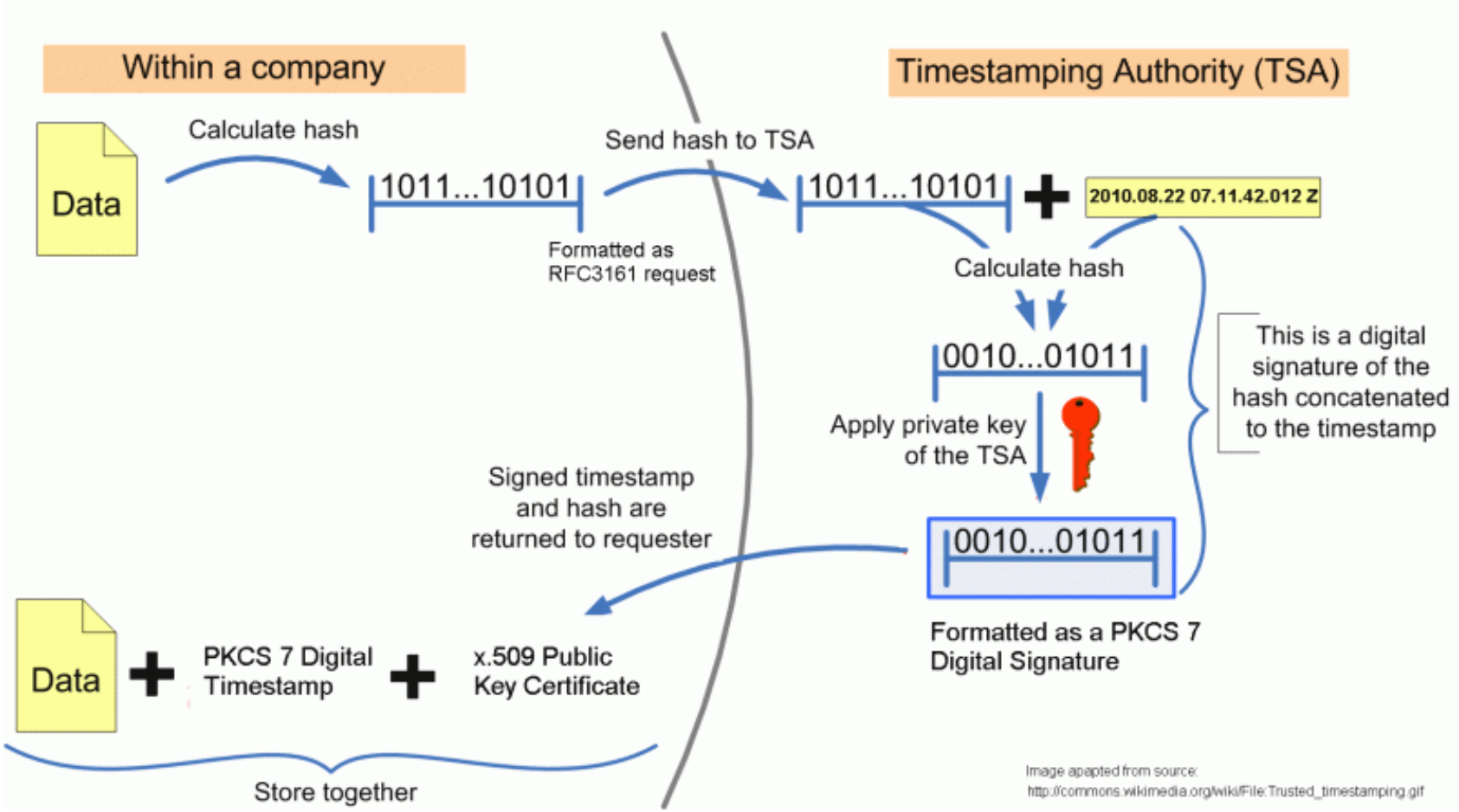
# Timestamp in OpenSSL
## (RFC 3161)

- Il Time-Stamp Protocol opera nel modo seguente
  1. Il client TSA calcola un valore hash per un certo file e lo invia alla TSA
  2. La TSA concatena la data e l'ora corrente al valore hash ricevuto, li firma ed invia il *timestamp* al client
     - Con la creazione del timestamp, la TSA certifica l'esistenza del file al momento della generazione della risposta
  3. Il client TSA
     - Riceve il timestamp e verifica la relativa firma
     - Controlla se il timestamp contiene lo stesso valore hash inviato alla TSA

OpenSSL
Cryptography and SSL/TLS Toolkit

# Timestamp in OpenSSL

## (RFC 3161)



# Timestamp in OpenSSL
## (Creazione della TSA)

- D'ora in avanti assumeremo che
  - **cacert.pem** sia il certificato della *CA*
  - **tsacert.pem** sia il certificato di firma della TSA, rilasciato dalla *CA*
  - **tsakey.pem** sia la chiave privata della TSA

OpenSSL
Cryptography and SSL/TLS Toolkit

# Timestamp in OpenSSL
## (Creazione della TSA)

- Per implementare una TSA è necessario
  - Creare la sua struttura
    - Directory, file, etc
  - Creare il suo certificato di firma
    - Tale certificato deve contenere particolari estensioni
    - N.B. Andranno specificati alcuni parametri nel file **`openssl.cnf`** sia della TSA che della CA che dovrà rilasciare il certificato di firma

OpenSSL
Cryptography and SSL/TLS Toolkit

# Timestamp in OpenSSL

## (Creazione della TSA)

1) Per creare una TSA è innanzitutto necessario creare una serie di directory e file, in maniera simile a quanto fatto per la creazione della CA

```
mkdir TSA
cd TSA
mkdir private
chmod g-rwx,o-rwx private
echo '01' > tsaserial
```

2) È inoltre necessario aggiungere la seguente estensione al file **openssl.cnf** sia della CA che della TSA

```
[v3_tsa]
basicConstraints=CA:FALSE
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer
keyUsage = nonRepudiation,digitalSignature
extendedKeyUsage = critical,timeStamping
```

# Timestamp in OpenSSL

## (Creazione della TSA)

1) Pe
direc

- OpenSSL permette di creare certificati per usi specifici, ad es., per una TSA
- Tali certificati vengono richiesti e creati specificando opportune opzioni (chiamate *estensioni*)
- Alcune di queste opzioni vanno specificate nel file openssl.cnf, altre mediante opzioni dei comandi per la richiesta e la creazione dei certificati (maggiori dettagli in seguito)
- v3_tsa è l'estensione che riguarda la Timestamping Authority
- Per maggiori informazioni: **`man x509v3_config`**

2) È inoltre nece ggiungere la seguente estensione al file **`openssl.cnf`** s lla CA che della TSA

```
[v3_tsa]
basicConstraints=CA:FALSE
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer
keyUsage = nonRepudiation,digitalSignature
extendedKeyUsage = critical,timeStamping
```

# Timestamp in OpenSSL

## (Creazione della TSA)

3) È inoltre necessario aggiungere i seguenti parametri al file **openssl.cnf** della TSA

```
[ tsa ]

default_tsa = tsa_config1

[ tsa_config1 ]
dir         = .
serial          = $dir/tsaserial
crypto_device    = builtin
signer_cert    = $dir/tsacert.pem
signer_key    = $dir/private/tsakey.pem
default_policy    = 1.2.3.4
digests         = md5, sha1
```

OpenSSL
Cryptography and SSL/TLS Toolkit

# Timestamp in OpenSSL

## (Creazione della TSA)

3) È inoltre necessario aggiungere i seguenti parametri al file **openssl.cnf** della TSA

```
[ tsa ]

default_tsa = tsa_config1

[ tsa_config1 ]
dir             = .
serial          = $dir/tsaserial
crypto_device    = builtin
signer_cert     = $dir/tsacert.pem
signer_key      = $dir/private/tsakey.pem
default_policy    = 1.2.3.4
digests         = md5, sha1
```

Root directory della TSA

# Timestamp in OpenSSL
## (Creazione della TSA)

3) È inoltre necessario aggiungere i seguenti parametri al file **openssl.cnf** della TSA

```
[ tsa ]

default_tsa = tsa_config1

[ tsa_config1 ]
dir         = .
serial          = $dir/tsaserial
crypto_device    = bui tin
signer_cert    = $dir/tsa t.pem
signer_key    = $dir/privat key.pem
default_policy    = 1.2.
digests         = md5, sh
```

Numero seriale corrente relativo alle risposte generate dalla TSA

# Timestamp in OpenSSL

## (Creazione della TSA)

3) È inoltre necessario aggiungere i seguenti parametri al file **openssl.cnf** della TSA

```
[ tsa ]

default_tsa = tsa_config1

[ tsa_config1 ]
dir         = .
serial          = $dir/tsaserial
crypto_device     = builtin
signer_cert     = $dir/tsacert.pem
signer_key     = $dir/private/tsakey.pem
default_policy     = 1.2.3.4
digests         = md5, sha1
```

OpenSSL engine usata per la firma

OpenSSL
Cryptography and SSL/TLS Toolkit

# Timestamp in OpenSSL
## (Creazione della TSA)

3) È inoltre necessario aggiungere i seguenti parametri al file **openssl.cnf** della TSA

```
[ tsa ]

default_tsa = tsa_config1

[ tsa_config1 ]
dir         = .
serial          = $dir/tsaserial
crypto device     = builtin
signer_cert     = $dir/tsacert.pem
signer_key     = $dir/private/tsakey.pem
default_policy     = 1.2.3.4
digests         = md5, sha1
```

Certificato usato dalla TSA per la firma

OpenSSL
Cryptography and SSL/TLS Toolkit

# Timestamp in OpenSSL
## (Creazione della TSA)

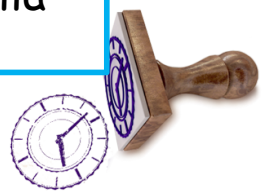
3) È inoltre necessario aggiungere i seguenti parametri al file **openssl.cnf** della TSA

```
[ tsa ]

default_tsa = tsa_config1

[ tsa_config1 ]
dir         = .
serial          = $dir/tsaserial
crypto_device    = builtin
signer_cert    = $dir/tsacert.pem
signer_key    = $dir/private/tsakey.pem
default_policy    = 1.2.3.4
digests         = md5, sha1
```

Chiave privata della TSA



OpenSSL
Cryptography and SSL/TLS Toolkit

# Timestamp in OpenSSL
## (Creazione della TSA)

3) È inoltre necessario aggiungere i seguenti parametri al file **openssl.cnf** della TSA

```
[ tsa ]

default_tsa = tsa_config1

[ tsa_config1 ]
dir             = .
serial          = $dir/
crypto_device    = built
signer_cert     = $dir/
signer_key      = $dir/private/tsakey.pem
default_policy     = 1.2.3.4
digests         = md5, sha1
```

Policy di default da usare quando nessuna policy è espressamente specificata nella richiesta

# Timestamp in OpenSSL
## (Creazione della TSA)

3) È inoltre necessario aggiungere i seguenti parametri al file **openssl.cnf** della TSA

```
[ tsa ]

default_tsa = tsa_config1

[ tsa_config1 ]
dir         = .
serial          = $dir/tsa
crypto_device    = built
signer_cert    = $dir/tsace
signer_key    = $dir/priv /tsakey.pem
default_policy    = 1.2.3.4
digests         = md5, sha1
```

Algoritmi supportati per il calcolo del message digest

OpenSSL
Cryptography and SSL/TLS Toolkit

# Timestamp in OpenSSL

## (Creazione Chiavi e Certificato della TSA)

1) La TSA genera la richiesta di certificato e la invia alla CA

```
openssl req -new -out tsa-req.pem -keyout private/tsakey.pem
-days 365 -extensions v3_tsa -config ./openssl.cnf
```

2) La CA genera il certificato per la TSA e lo invia a quest'ultima

```
openssl ca -out tsacert.pem -extensions v3_tsa
-config ./openssl.cnf -in tsa-req.pem
```

# Timestamp in OpenSSL
## (Comandi Coinvolti)

- Il comando **ts** permette di implementare in OpenSSL tutte le funzionalità di una TSA, fornendo tre funzioni principali
  1. Creazione di una *timestamp request* relativa ad un file
     - **openssl ts -query**
  2. Creazione di una *timestamp response* in base ad una richiesta
     - **openssl ts -reply**
  3. Verifica della corrispondenza tra una risposta ed una particolare richiesta (o un determinato file)
     - **openssl ts -verify**

OpenSSL
Cryptography and SSL/TLS Toolkit

# Timestamp in OpenSSL

## (Creazione di una Richiesta – Timestamp Request)

### Struttura generale del comando `ts -query`

```
openssl ts -query args
```

- **args**
  - **-data file_to_hash** File per il quale deve essere creata la richiesta di timestamp
  - **-config file** File di configurazione
  - **-digest digest_bytes** Permette di usare direttamente il digest, senza specificare il file coi dati
  - **-md2|-md4|-md5|-sha|-sha1|-mdc2|-ripemd160|...** Message digest da applicare al file. Possono essere usati tutti gli algoritmi supportati da OpenSSL
  - **-no_nonce** Nessun *nonce* viene specificato nella richiesta. Altrimenti, è incluso nella richiesta un *nonce* pseudorandom di 64 bit
  - **-cert** Richiede alla TSA di includere nella risposta il relativo certificato di firma
  - **-in request.tsq** Richiesta di timestamp precedentemente creata in formato DER
  - **-out request.tsq** File di output in cui verrà memorizzata la richiesta
  - **-text** Restituisce l'output in formato human-readable

# Timestamp in OpenSSL

## (Creazione di una Richiesta – Timestamp Request)

### Struttura generale del comando `ts -query`

```
openssl ts -query args
```

- **args**
  - **-data file_** sere creata la richiesta di timestamp
  - **-config fil**
  - **-digest digest_bytes** Permette di usare direttamente il digest, senza specificare il file coi dati
  - **-md2|-md4|-md5|-sha|-sha1|-mdc2|-ripemd160|...** Message digest da applicare al file. Possono essere usati tutti gli algoritmi supportati da OpenSSL
  - **-no_nonce** Nessun *nonce* viene specificato nella richiesta. Altrimenti, è incluso nella richiesta un *nonce* pseudorandom di 64 bit
  - **-cert** Richiede alla TSA di includere nella risposta il relativo certificato di firma
  - **-in request.tsq** Richiesta di timestamp precedentemente creata in formato DER
  - **-out request.tsq** File di output in cui verrà memorizzata la richiesta
  - **-text** Restituisce l'output in formato human-readable

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

# Timestamp in OpenSSL

## (Esempio di Timestamp Request)

Mediante il seguente comando è possibile creare una richiesta di timestamp per **file1.txt**

- Viene usato SHA-1
- Non viene inviato un nonce
- Non è richiesto il certificato della TSA nella risposta

```
openssl ts -query -data file1.txt -no_nonce -out file1.tsq
```

Mediante il seguente comando è possibile creare una richiesta di timestamp simile alla precedente, ma specificando esplicitamente il digest del messaggio

```
openssl ts -query -digest
b7e5d3f93198b38379852f2c04e78d73abdd0f4b -no_nonce
-out file1.tsq
```

OpenSSL Cryptography and SSL/TLS Toolkit

# Timestamp in OpenSSL

## (Esempio di Timestamp Request)

Mediante il seguente comando è possibile visualizzare la precedente richiesta, contenuta in **file1.tsq**

```
openssl ts -query -in file1.tsq -text
```

```
Version: 1
Hash Algorithm: sha1
Message data:
    0000 - 33 36 31 fd f9 2f b8 32-7e 08 67 e9 c0 64 ce 07   361../.2~.g..d..
    0010 - ac d7 2c 26                                       ..,&
Policy OID: unspecified
Nonce: unspecified
Certificate required: no
Extensions:
```

OpenSSL Cryptography and SSL/TLS Toolkit

# Timestamp in OpenSSL

## (Esempio di Timestamp Request)

Mediante il seguente comando è possibile creare una richiesta di timestamp per l'MD5 di **file2.txt**

- Nella richiesta è incluso un nonce
- La TSA dovrà includere il proprio certificato nella risposta

```
openssl ts -query -data file2.txt -md5 -cert -out file2.tsq
```

Visualizzando il contenuto di **file2.tsq** è possibile notare come i suddetti parametri siano inclusi nella richiesta

```
Version: 1
Hash Algorithm: md5
Message data:
    0000 - 1a a0 97 55 4d 00 ea 21-d4 51 d2 99 ab c7 e8 bd   ...UM..!.Q......
Policy OID: unspecified
Nonce: 0x0E46C82A9F21147F
Certificate required: yes
Extensions:
```

OpenSSL Cryptography and SSL/TLS Toolkit

# Timestamp in OpenSSL

## (Creazione di una Risposta – Timestamp Response)

### Struttura generale del comando `ts -reply`

```
openssl ts -reply args
```

- **args**
  - **`-queryfile request.tsq`** File contenente la richiesta di timestamp in formato DER
  - **`-signer tsa_cert.pem`** Certificato della TSA in formato PEM
  - **`-inkey private.pem`** Chiave privata della TSA in formato PEM
  - **`-in response.tsr`** Specifica un response timestamp o un timestamp token precedentemente creato in formato DER
  - **`-out response.tsr`** La risposta della TSA è scritta in questo file
  - **`-token_out`** L'output è un timestamp token invece che un response timestamp

OpenSSL Cryptography and SSL/TLS Toolkit

# Timestamp in OpenSSL

## (Esempio di Timestamp Response)

Mediante il seguente comando è possibile creare una risposta (timestamp response) ad una richiesta di timestamp

```
openssl ts -reply -config ./openssl.cnf -queryfile file1.tsq
-inkey private/tsakey.pem -signer tsacert.pem -out file1.tsr
```

Mediante il seguente comando è possibile visualizzare il contenuto di una timestamp response

```
openssl ts -reply -in file1.tsr -text
```

# Timestamp in OpenSSL

## (Timestamp Response vs. Timestamp token)

Timestamp response

```
Status info:
Status: Granted.
Status description: unspecified
Failure info: unspecified
```

Timestamp token

```
TST info:
Version: 1
Policy OID: 1.2.3.4
Hash Algorithm: sha1
Message data:
    0000 - 33 36 31 fd f9 2f b8 32-7e 08 67 e9 c0 64 ce 07   361../.2~.g..d..
    0010 - ac d7 2c 26                                       ..,&
Serial number: 0x03
Time stamp: Mar  1 14:06:04 2017 GMT
Accuracy: unspecified
Ordering: no
Nonce: unspecified
TSA: unspecified
Extensions:
```

Timestamp response contenuta nel file **file1.tsr**

# Timestamp in OpenSSL

## (Esempio di Timestamp Response)

Mediante il seguente comando è possibile creare direttamente un timestamp token invece di una timestamp response

```
openssl ts -reply -config ./openssl.cnf -queryfile file1.tsq
-out file1_token.der -token_out
```

Mediante il seguente comando è possibile estrarre il timestamp token da una timestamp response

```
openssl ts -reply -in file1.tsr -out file1_token.der -token_out
```

OpenSSL
Cryptography and SSL/TLS Toolkit

# Timestamp in OpenSSL

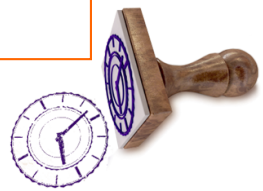
## (Esempio di Timestamp Response)

Mediante il seguente comando è possibile visualizzare il contenuto del timestamp token

```
openssl ts -reply -config ./openssl.cnf -in file1_token.der
-token_in -text -token_out
```

```
Version: 1
Policy OID: 1.2.3.4
Hash Algorithm: sha1
Message data:
    0000 - 33 36 31 fd f9 2f b8 32-7e 08 67 e9 c0 64 ce 07   361../.2~.g..d..
    0010 - ac d7 2c 26                                       ..,&
Serial number: 0x03
Time stamp: Mar  1 14:06:04 2017 GMT
Accuracy: unspecified
Ordering: no
Nonce: unspecified
TSA: unspecified
Extensions:
```

OpenSSL Cryptography and SSL/TLS Toolkit



# Timestamp in OpenSSL

## (Esempio di Timestamp Response)

Mediante il seguente comando è possibile creare una timestamp response a partire da un timestamp token

```
openssl ts -reply -in design1_token.der -token_in -out design1.tsr
```

Contenuto del file **design1.tsr**

```
Status info:
Status: Granted.
Status description: unspecified
Failure info: unspecified

TST info:
Version: 1
Policy OID: 1.2.3.4
Hash Algorithm: sha1
Message data:
    0000 - 33 36 31 fd f9 2f b8 32-7e 08 67 e9 c0 64 ce 07   361../.2~.g..d..
    0010 - ac d7 2c 26                                       ..,&
Serial number: 0x03
Time stamp: Mar  1 14:06:04 2017 GMT
Accuracy: unspecified
Ordering: no
Nonce: unspecified
TSA: unspecified
Extensions:
```

# Timestamp in OpenSSL
## (Verifica)

### Struttura generale del comando ts

```
openssl ts -verify args
```

- **args**
  - **-data file_to_hash** Risposta (o token) che deve essere verificata rispetto a file_to_hash
    - È calcolato l'hash del file mediante l'algoritmo specificato nel token
  - **-in response.tsr** Response timestamp, in formato DER, che deve essere verificato
  - **-CAfile trusted_certs.pem** File contenente un insieme di certificati trusted e self-signed, in formato PEM
  - **-untrusted cert_file.pem** File contenente un insieme di certificati non trusted, in formato PEM, che possono essere necessari alla costruzione della *certificate chain* relativa al certificato di firma della TSA
  - **-queryfile request.tsq** Richiesta di timestamp in formato DER

OpenSSL Cryptography and SSL/TLS Toolkit

# Timestamp in OpenSSL

## (Esempio di Timestamp Verify)

Mediante il seguente comando è possibile verificare una timestamp response relativa ad una richiesta

```
openssl ts -verify -config ./openssl.cnf -queryfile file1.tsq -in
file1.tsr -CAfile cacert.pem -untrusted tsacert.pem
```

Mediante il seguente comando è possibile verificare una timestamp response rispetto ad una richiesta che include la certificate chain

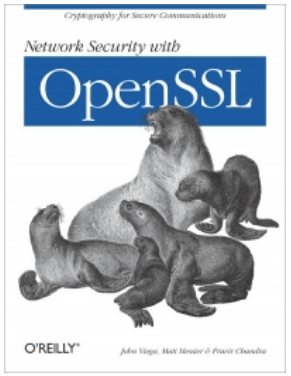
```
openssl ts -verify -config ./openssl.cnf -queryfile file2.tsq -in
file2.tsr -CAfile cacert.pem
```

Mediante il seguente comando è possibile verificare una timestamp response rispetto al file originario

```
openssl ts -verify -config ./openssl.cnf -data file2.txt -in
file2.tsr -CAfile cacert.pem
```

# Bibliografia

- Network Security with OpenSSL
  Pravir Chandra, Matt Messier and John Viega (2002), O'Reilly
  - Cap. 3
  - Appendix A. Command-Line Reference




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

OpenSSL
Cryptography and SSL/TLS Toolkit

# Domande?