

Generazione di certificati X.509 nell'ambito dei
test di interoperabilità dei Sistemi di Posta
Elettronica Certificata (PEC)

Antonio De Maglio

15 ottobre 2014

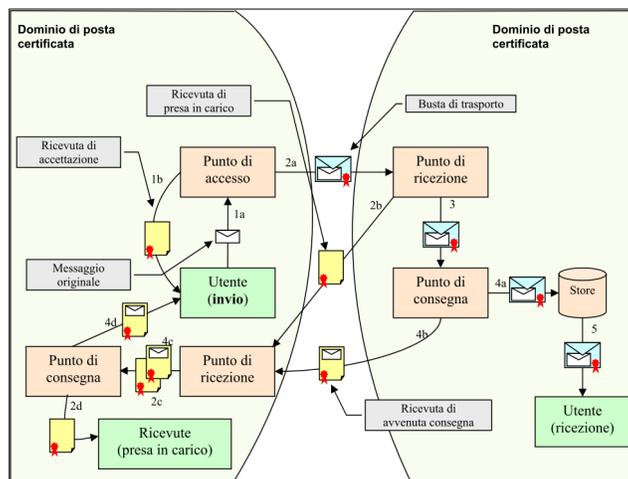


Figura 1: funzionamento PEC

1 Introduzione

L'Istituto di Scienza e Tecnologie dell'Informazione (ISTI) ha un contratto con Agenzia per l'Italia Digitale (AgID) per effettuare i test di interoperabilità della Posta Elettronica Certificata (PEC).

In figura 1 possiamo vedere uno schema sul funzionamento della pec, ulteriori informazioni sulla PEC possono essere trovati nell'RFC6109. [IET11] Questi test di interoperabilità hanno lo scopo di verificare che ogni gestore PEC sia pienamente conforme a quanto stabilito dalle "Regole tecniche del servizio di trasmissione di documenti informatici mediante posta elettronica certificata". Maggiori informazioni sui test possono essere trovate in: [MBV12].

L'ISTI quindi è dotato sia di un proprio gestore PEC, che da una piattaforma per l'esecuzione di una batteria di test sul gestore da verificare.

La nostra piattaforma prevede che in fase di configurazione di una sessione di test per un gestore, se non già presenti, vengano generati automaticamente i certificati X.509 delle caselle di posta elettronica che vengono utilizzate nella batteria di test corrente.

Descriverò in questa guida inizialmente i comandi openssl eseguiti per generare una nostra Certification Authority (CA), successivamente analizzerò i comandi openssl per generare i certificati degli utenti.

Terminerò questa guida illustrando i passi per revocare un certificato emesso dalla propria CA.

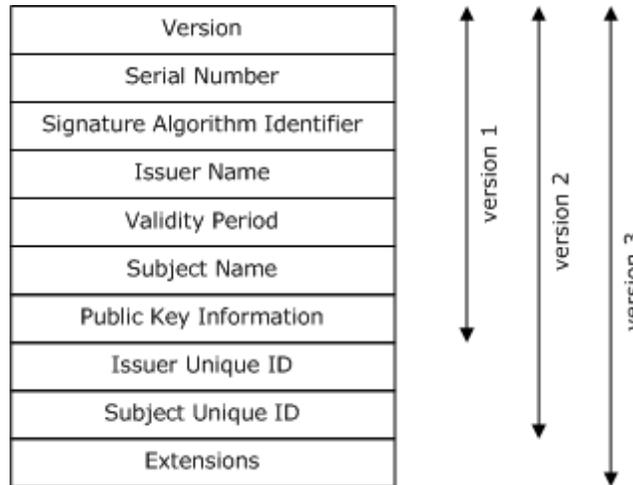


Figura 2: schema di certificato X.509

2 Generazione dei certificati X.509 di una propria Certification Authority

Inizialmente bisognerà generare il certificato X.509 della nostra Certification Authority(CA) che utilizzeremo per emettere i certificati dei nostri utenti.

In questo lavoro la CA userà un certificato autofirmato, i test di interoperabilità vengono svolti solamente da personale ISTI, quindi possiamo fidarci della nostra CA. Se così non fosse dovremmo farci emettere i certificati da una CA ufficiale riconosciuta oppure accreditarci come CA.

Il primo passo consiste nel generare una coppia di chiavi: una pubblica e una privata. La chiave privata va messa in un luogo sicuro, mentre la chiave pubblica può essere inserita in un sito internet o mandata via mail.

La chiave privata della nostra CA verrà utilizzata per firmare i certificati emessi ai nostri utenti, la chiave pubblica verrà utilizzata dai nostri utenti per verificare se un certificato è stato emesso veramente dalla nostra CA. Di seguito il comando openssl per generare la coppia di chiavi suddetta:

```
$openssl req -nodes -config openssl.cnf -days 1095 -x509 -sha1 -newkey
rsa:2048 -out root.pem -outform PEM
```

- nodes: la chiave privata sarà memorizzata in chiaro.
- config: permette di specificare un file in cui sono inserite tutte le informazioni per emettere il certificato, altrimenti (non utilizzabile per i nostri scopi) verrebbero richieste in modalità interattiva.
- days: per quanti giorni sarà valido il certificato

- x509: il comando openssl restituirà un certificato X.509. L'altra opzione (come vedremo in seguito) prevede che il comando openssl restituisca una richiesta di certificato(request) che va inviata ad una CA. Sarà la CA, quindi ad emettere il certificato firmandolo con la propria chiave privata.
- newkey: specifica che deve essere generata una nuova coppia di chiavi pubblica/privata
- out: specifica il file in cui deve essere salvata la chiave pubblica
- outform: specifica il formato del file di output. Può essere Distinguished Encoding Rules(DER) o Privacy-enhanced Electronic Mail(PEM).

In figura 2 possiamo vedere uno schema di certificato X.509.
Di seguito riporto il file openssl.cnf utilizzato:

```
[ req ]
default_bits           = 2048
default_keyfile        =
postacert_root:[ test.certs.private]root.pem
default_md             = sha512
prompt                 = no
distinguished_name     = root_ca_distinguished_name
x509_extensions = v3_ca
[ root_ca_distinguished_name ]
countryName            = IT
stateOrProvinceName   = Pisa
localityName           = Pisa
0.organizationName    = ISTI-CNR
commonName             = ISTC Certification Authority
[ v3_ca ]
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid:always,issuer:always
basicConstraints = CA:true
[ ca ]
default_ca             = CA_default

[ CA_default ]
dir                    = POSTACERT_ROOT:[TEST.CERTS]

new_certs_dir          = POSTACERT_ROOT:[TEST.CERTS.signed-keys]
database               = POSTACERT_ROOT:[TEST.CERTS.root.conf]index
certificate            = POSTACERT_ROOT:[TEST.CERTS.public]root.pem
serial                 = POSTACERT_ROOT:[TEST.CERTS.conf]serial
private_key            = POSTACERT_ROOT:[TEST.CERTS.private]root.pem
x509_extensions       = usr_cert
default_md             = sha1
policy                 = policy_match
```

```
[ policy_match ]
countryName           = optional
stateOrProvinceName  = optional
organizationName      = optional
organizationalUnitName = optional
commonName            = optional
emailAddress          = optional
```

```
[ usr_cert ]
basicConstraints=CA:FALSE
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid , issuer : always
```

Commentiamo ora i vari parametri del file openssl.cnf:

- default_bits: indica il numero di bit di default della chiave privata
- default_keyfile: indica il file dove viene memorizzata la chiave privata
- default_md: indica il metodo di default con cui verterà firmato il certificato, i metodi permessi sono:
 - d4
 - md5
 - ripemd160
 - sha
 - sha1
 - sha224
 - sha256
 - sha384
 - sha512
 - whirlpool
- prompt: indica se possono essere richiesti a prompt alcuni parametri
- distinguished_name: specifica la sezione in cui inseriremo alcune informazioni sulla nostra CA come lo stato, la provincia, il nome della nostra organizzazione e così via.
- x509_extensions: indica la sezione in cui inseriremo dei valori per l'estensione di openssl che indica gli scopi per cui può essere utilizzato il certificato.
- subjectKeyIdentifier: è costituito da un hash della chiave pubblica
- authorityKeyIdentifier: Identifica la CA.

- `basicConstraints = CA:true` identifica che il certificato X.509 è stato emesso per una CA
- `new_certs_dir`: directory che contiene i certificati emessi dalla CA
- `database`: indica il database dei certificati emessi
- `certificate`: indica la chiave pubblica della CA
- `serial`: indica il prossimo numero seriale
- `private_key`: indica la chiave privata della CA
- `policy`: indica la sezione in cui per ogni campo del certificato dell'utente si indica se è obbligatorio

Dopo aver emesso il certificato della CA, se vogliamo visualizzare in formato testo il certificato possiamo eseguire il comando:

```
$ openssl x509 -in cert.pem -noout -text
```

Analizziamo i parametri:

- `in`: il nome del file che contiene il certificato in formato PEM
- `noout`: la request non viene stampata codificata
- `text`: stampa il certificato in formato testo

3 Creazione di certificati firmati dalla nostra Certification Authority

Avendo creato il certificato della nostra CA possiamo richiederle di firmare i certificati X.509 dei nostri utenti.

3.1 Generazione del certificato X.509 di un utente

In questa sezione analizzeremo i passi affinché l'utente possa avere un certificato X.509 emesso dalla CA precedentemente creata.

L'utente dovrà creare la coppia di chiavi pubblica e privata, la chiave privata verrà usata per firmare e cifrare i propri messaggi e deve essere protetta; la chiave pubblica invece può essere inviata via mail o anche inserita in un proprio sito Internet.

Per generare la coppia di chiavi pubblica e privata l'utente utilizzerà il seguente comando:

```
$ openssl genrsa -out 'key' 2048
```

Per poter generare la richiesta (request) che come suddetto l'utente dovrà inviare alla propria CA si utilizza il seguente comando:

```
$openssl req -new -key key.key -config openssl.cnf -out request.csr '
```

Un esempio di openssl.cnf per l'utente è:

```
[req]
default_bits = 2048
default_md = sha1
prompt = no
distinguished_name = root_ca_distinguished_name
[root_ca_distinguished_name]
countryName = IT
stateOrProvinceName = Pisa
localityName = Pisa
0.organizationName = utente pec
commonName = email
emailAddress= email
```

La CA ricevuta la richiesta di certificato dall'utente provvederà ad emettere il certificato da inviare all'utente mediante il comando:

```
$openssl ca -batch -config openssl.cnf -in request.csr -out certificate.crt
```

Analizziamo i parametri non specificati precedentemente:

- batch: indica che il certificato verrà emesso automaticamente e nessuna domanda verrà richiesta a prompt
- in: il nome del file che contiene la request
- out: il file in cui viene salvato il certificato in formato CRT

La CA invierà quindi all'utente il certificato emesso che potrà inserire nel proprio client di posta elettronica o webmail per cifrare e/o firmare i messaggi e-mail.

3.2 Generazione automatica dei certificati di una serie di utenti e integrazione nel sistema pec ISTI

A partire dai comandi illustrati sopra ho scritto alcuni script per automatizzare la gestione dei certificati integrandoli in pectestmanager. Pectestmanager permette di inviare uno o più test per un determinato gestore. Questi script sono:

- genall.com: genera la CA ed emette i certificati per tutti gli utenti.
- genca.com: genera la CA
- genonecert.com: genera il certificato di un utente. Nel momento in cui si crea un gestore in pectestsmanager o si modifica l'indirizzo e-mail da cui vengono inviate le mail viene invocato:

```
$ generateonecert P1
```

P1 indica l'indirizzo e-mail dell'utente che si vuole emettere il certificato; se il certificato è stato già emesso, non viene generato un altro.

- generateallcertificate.com: emette i certificati di tutti gli utenti leggendo gli indirizzi e-mail dai profili dei gestori pec memorizzati tramite pectestshistory e pectestmanager.

4 Revoca di certificati di un utente

Per revocare il certificato di un utente la CA userà:

```
openssl ca -revoke serial.pem
```

Openssl memorizza ogni certificato emesso nella cartella newcerts, l'installazione di openssl presente sul sistema pec ISTI le memorizza nella cartella signed-keys.

Riferimenti bibliografici

- [IET11] IETF. Rfc6109 - la posta elettronica certificata - italian certified electronic mail. <http://tools.ietf.org/html/rfc6109>, 2011.
- [MBV12] C. Petrucci M. Buzzi, F. Gennai and A. Vinciarelli. E-government services: Quality assurance of the italian certified electronic mail. In *E-government services: Quality Assurance of the Italian Certified Electronic Mail, HCITOCH 2012, Venezia, 2012*.
- [pro] Openssl project. Openssl documents. <https://www.openssl.org/docs/apps/openssl.html>.