

## Esempio di certificato digitale con OpenSSL

Di seguito si riporterà un esempio su come si creano praticamente certificati digitali. Si utilizzerà software open source (disponibile per sistemi Windows e Linux), OpenSSL (la versione windows scaricabile dal sito [www.slproweb.com/products/Win32OpenSSL.html](http://www.slproweb.com/products/Win32OpenSSL.html)) versione 0.9.8e Light.. Dopo aver scaricato il pacchetto (Win32OpenSSL\_Light-0\_9\_8e.exe) e averlo installato è necessario eseguire la shell del DOS poiché l' applicativo funziona da riga di comando. Dal prompt entrare in c:\openssl\bin e digitare openssl per entrare sulla riga di comando di OpenSSL (su alcuni sistemi potrebbe essere necessario installare la libreria msvc71.dll, scaricabile da internet, che deve essere installata in c:\windows\system32).

Il 90-95 % delle CA si basano su OpenSSL. Infatti ca.pl (che si trova dentro c:\openssl\bin) è un programma scritto in Perl che viene automaticamente installato con OpenSSL e che permette di generare la propria chiave privata PKI, generare richieste di certificati e di firmarli e perfino revocarli da un archivio precedentemente generato: è possibile cioè creare una piccola CA che funziona benissimo, se gestita accuratamente, anche con qualche migliaio di utenti. Questo pacchetto ha avuto ultimamente anche una certificazione americana, FIPS livello 3 (organismo di certificazione del governo americano, riconosciuto anche a livello europeo, ad esempio a livello dei common criteria): questo vuol dire che seguendo opportune procedure, è possibile creare una Certification Authority in modo certificato, cioè in un modo che è riconosciuto dalla legge.

### Generazione certificati firmati Versione 1

Digitando:

```
OpenSSL>openssl ?
```

si ha una lista di possibili comandi e gli algoritmi di cifratura simmetrica e asimmetrica supportati. Il primo comando che vedremo è l'x509, di particolare importanza perché i certificati digitali seguono lo standard x509 versione 3. Il primo passo da seguire per creare un certificato è generare la chiave privata dalla quale poi ricavare la chiave pubblica e tutte le altre informazioni.

Per generare una chiave privata di 4096 bit che chiamiamo key\_0 si esegue il seguente comando:

```
OpenSSL>genrsa -out key_0.pem 4096
```

Il formato pem è un formato di scambio di informazioni. L' opzione -out scrive la chiave così generata nel file key\_0.pem. Così facendo richiediamo che il certificato abbia chiavi generate tramite RSA. Notiamo che il tempo di generazione della chiave richiede qualche secondo: la cifratura asimmetrica è in generale pesante. Si consideri che se queste operazioni vengono fatte ad esempio da un sito web tramite https, la cifratura rallenta di molto le operazioni delle transizioni.

Per vedere la chiave così generata, digitare exit per tornare al prompt di MS-DOS e digitare type key\_0.pem. Si può notare che le righe sono tutte uguali e composte da caratteri minuscoli, maiuscoli, numeri, \ e + o -: si è in presenza di un codificatore base 64. Il base 64 è utilizzato per convertire in numeri binari una sequenza di caratteri ASCII standard leggibili (e quindi trasmissibili sulla rete), aumentando però le dimensioni (in genere il rapporto è 2/3).

La chiave anche se in ASCII risulta comunque illeggibile, perlomeno nella sua struttura. Per capire come è strutturata la chiave bisogna digitare:

```
OpenSSL>rsa -in key_0.pem -text -noout
```

ciò traduce in formato testo la chiave generate nel file key\_0.pem secondo RSA, senza generare in uscita dati. Il layout adesso mostra una prima parte che è un modulo a 4096 bit scritto in esadecimale; poi un esponente pubblico (un piccolo numero dispari, in questo caso 65537, ma in certe chiavi generate su smart card può essere 3 o 7, cioè numeri molto piccoli); successivamente l' esponente privato e i due numeri primi che moltiplicati tra di loro ci danno il modulo: questi rappresentano la vera chiave, entrambi a 2048 bit. In ultimo gli esponenti e il coefficiente rappresentano dei valori utilizzati dall' algoritmo per accelerare il processo di cifratura. Dobbiamo ora generare un certificato con questa coppia dei chiavi pubblica/privata. Ciò si fa tramite il comando:

```
OpenSSL>req -new -key key_0.pem -out req0.pem
```

si effettua una richiesta di un nuovo certificato, cioè di un documento che è la prova che si è in possesso di una chiave privata e di altre informazioni, come per esempio il nome, cognome, e così via. All' invio del comando viene richiesto in sequenza (attributi di primo livello, cioè le policy di sicurezza scritte nel file openssl.cnf: modificando questo file, cambiano le richieste):

- a) *Country name*: è il nome dello stato di appartenenza (es: Italia);
- b) *State or Province Name*: è il nome della regione (es: Lazio)
- c) *Locality Name*: è il nome della città (es: Roma)
- d) *Organization Name*: nome della organizzazione (es: Università Tor Vergata)
- e) *Organizational Unit Name*: è l' unità organizzativa di riferimento (es: Master)
- f) *Common Name*: l' identificativo finale del proprietario del certificato (es: root CA, essendo quella che stiamo creando una CA che certifica altri certificati)
- g) *Email Address*: è l' indirizzo email, facoltativo (in genere per una root CA va messo per segnalare al gestore eventuali malfunzionamenti)
- h) *A challenge password, an optional company name*: attributi extra (trascurabili per i nostri scopi)

La richiesta così generata può essere visualizzata digitando type req0.pem. Per visualizzarlo in un formato leggibile si digita:

```
OpenSSL>req -in req0.pem -text -noout
```

In testa c' è la versione del certificato (versione 0 perché ancora non è un certificato, ma deve essere firmato dalla CA) e il subject (cioè l' insieme delle informazioni che distinguono il proprietario, quelle appena immesse secondo una nomenclatura derivata dagli elenchi telefonici).

Il common name è quello più importante perché è quello che viene scambiato a livello di comunicazione SSL con il Web Server: il proprietario del Web Server conosce il nostro common name, ancora prima della richiesta del servizio.

Inoltre si può vedere che l' algoritmo di crittografia usato è l' RSA e la public key è di 4096 bit e l' esponente 65537. L' algoritmo di firma è SHA1 (ultimi bit del certificato): ciò è utilizzato per dimostrare che si è in possesso della chiave privata e della chiave pubblica fornita nel certificato.

Si ricordi infatti che quando viene generata una richiesta si eseguono i seguenti passi:

- ? fornire una chiave pubblica:  $k_{pub}$
- ? cifrare lo SHA1 della chiave pubblica con la chiave privata:  $k_{pri}(SHA1(k_{pub}))$

In ricezione:

- ? calcolare lo SHA1 della chiave pubblica  $k_{pub}$

- ? tale valore deve essere uguale alla decifratura con chiave pubblica della firma cifrata con chiave privata:  $SHA1(k_{pub}) = k_{pub}(k_{pri}(SHA1(k_{pub})))$

Quindi, adesso nella directory bin è presente sia il file key\_0.pem relativo alla chiave privata generata, sia il file req0.pem relativo alla richiesta.

Ogni certification authority firma un certificato. Poiché stiamo creando una root CA, dobbiamo firmarci noi il nostro certificato, tramite il seguente comando:

```
OpenSSL>x509 -req -in req0.pem -signkey key_0.pem -out cert0.pem
```

Per vedere il certificato così firmato è sufficiente dare il seguente comando:

```
OpenSSL>x509 -in cert0.pem -text -noout
```

Da qui è possibile vedere che il certificato firmato è diverso dalla richiesta, valido a tutti gli effetti, composto dai seguenti campi:

- a) *Version*: è il numero della versione. Oggi siamo alla versione 3 (il certificato che abbiamo creato riporta la versione 1 perché per la sua generazione abbiamo utilizzato solo una chiave privata, esiste un metodo per generarlo di tipo 3 che utilizza oltre alla chiave privata anche altri valori, vedere più avanti);
- b) *Serial Number*: unico per tutti i certificati
- c) *Signature Algorithm*: già dichiarato nella richiesta di certificato (SHA1)
- d) *Issuer*: un campo per tenere nota di chi ha emesso il certificato. Poiché la nostra è una root CA, lo abbiamo emesso noi stessi (autofirmandolo) e quindi questo campo è lo stesso del Subject (vedi dopo) che identifica il proprietario
- e) *Validità*: chi firma il certificato dà la validità. Per default è posta ad un mese

### Generazione certificati firmati Versione 3

La procedura è analoga a quella appena vista. Proviamo dapprima a creare un testo con notepad (chiamiamolo ad esempio prova.txt) contenente la seguente frase “Nel mezzo del cammin di nostra vita.” (esattamente, così senza introdurre spazi aggiuntivi) salvandolo dentro c:\openssl\bin e digitiamo:

```
OpenSSL>sha1 prova.txt
```

Il risultato è il seguente valore di Hash: 9157922ac5eb10b75b02f3462efe03ebf772334d.

Ovviamente anche solo cambiando un carattere del file prova.txt (ad esempio inserendo anche un carattere blank alla fine della riga) si ottiene tutto un altro valore di Hash. Questa è una stringa di 160 bit (40 caratteri esadecimale). Analogamente si può fare l' Hash con MD5 digitando:

```
OpenSSL>md5 prova.txt
```

Il risultato è il seguente valore di Hash: 4928baf884ac990f13e0cc968393d180. Questa è una stringa di 128 bit (32 caratteri esadecimale).

Un problema di sicurezza è che Hash e messaggio devono viaggiare insieme e devono essere strettamente correlati, cioè bisogna espressamente dichiarare che ad un determinato messaggio è associato un ben preciso Hash.

Come prima generiamo una chiave RSA:

```
OpenSSL>genrsa -out root_ca_key.pem 2048
```

a 2048 bit per la nostra root CA.

Il comando per generare il certificato firmato è:

```
OpenSSL>req -x509 -out root_ca_cert.pem -new -key root_ca_key.pem -days 3650 -  
set_serial 1000000
```

tale certificato è autofirmato, è valido 10 anni (3650 giorni) e ha 1000000 come numero seriale. Come prima, vengono richiesti alcuni valori:

- i) *Country name*: è il nome dello stato di appartenenza (es: Italia);
- j) *State or Province Name*: è il nome della regione (es: Lazio)
- k) *Locality Name*: è il nome della città (es: Roma)
- l) *Organization Name*: nome della organizzazione (es: Università)
- m) *Organizational Unit Name*: è l'unità organizzativa di riferimento (es: Master)
- n) *Common Name*: l'identificativo finale del proprietario del certificato (es: Mario Rossi)
- o) *Email Address*: è l'indirizzo email, facoltativo (es: MarioRossi@rossi.it)

Vediamo il contenuto del certificato così generato:

```
OpenSSL>x509 -text -noout -in root_ca_cert.pem
```

Questa volta il certificato è una versione 3: l'emittitore (issuer) è uguale al proprietario (subject) e ciò fa capire che è un certificato root.

Il certificato che abbiamo creato è per una certification authority che deve firmare certificati. Mancano però dei parametri che non sono stati generati perché nel file openssl.cnf non è stata introdotta l'estensione `keyUsage = cRLSign, keyCertSign` (se si apre openssl.cnf con notepad si vedrà che questa estensione è commentata e quindi non ha effetto): una certification authority deve poter firmare la lista di revoca dei certificati CRL e deve poter avere come attributo la firma delle chiavi dei propri certificati.

Vediamo adesso di importare il certificato così generato in Internet Explorer.

Bisogna andare in Strumenti ? Opzioni Internet ? Contenuto ? Certificati ? Importa. Il certificato deve essere importato dalla nostra applicazione (premere su avanti alla prima finestra, cliccare su sfoglia cercando il file nella directory `c:\openssl\bin\root_ca_cert.pem` (selezionando tutti i file dal menu Tipo file), confermare con avanti, alla finestra successiva spuntare "Selezionare automaticamente l'archivio certificati secondo il tipo di certificato", cliccare su avanti e nella finestra successiva fine e confermare sì alla domanda "installare il certificato?")

Per verificare che il certificato è stato installato, nella finestra certificati andare nella scheda "Autorità di certificazione fonti attendibili" (scorrere fino a trovare il nome e cognome del soggetto a cui è stato rilasciato, Mario Rossi nel nostro caso perché abbiamo messo questo common name).

**A cura di Antonio Guzzo**