Alma Mater Studiorum · Università di Bologna

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI
Corso di Laurea in Matematica

LA PKI

E L'ATTACCO

MAN IN THE MIDDLE

Tesi di laurea in Algoritmi della Teoria dei Numeri e Crittografia

Relatore:
Davide Aliffi

Presentata da:
Laura Mei

III Sessione
Anno Accademico 2014-2015

Introduzione

Al giorno d'oggi, con il vastissimo utilizzo della rete Internet, si va incontro alla sempre più crescente necessità di proteggere e mantenere segreti i propri dati personali e lo scambio di informazioni.
Scopo di questa tesi è lo studio di come l'Infrastruttura a chiave pubblica (PKI) possa essere violata tramite attacchi informatici che minano la privacy e la segretezza dei dati che circolano in Internet, in particolar modo attraverso il cosiddetto Man in the Middle, un fenomeno che negli ultimi anni è in notevole crescita, ed è in grado di sorpassare le misure di sicurezza che ad oggi vengono utilizzate sul Web.
Il lavoro è articolato in due fasi distinte.
Nella prima fase si introducono i due algoritmi crittografici più utilizzati per cifrare e decifrare i messaggi sul Web: si tratta dell'algoritmo RSA e Diffie-Hellman.
Nella seconda fase si introducono i vari tipi di minacce alla sicurezza e alla privacy che si incontrano sul Web, spiegando nel dettaglio il meccanismo di attacco MitM. Si andrà poi a definire il concetto di Infrastruttura a chiave pubblica, il cui scopo è quello di associare ad un utente un paio di chiavi, pubblica e privata, attraverso le quali potrà comunicare in forma criptata con altri utenti. Tale associazione chiave-utente viene resa ufficialmente valida da un certificato che contiene la firma di un'autorità fidata: l'Autorità di Certificazione (CA). Nell'ambito della sicurezza sul Web, la CA ricopre un ruolo di fondamentale importanza: è infatti proprio grazie ad essa che, attraverso il rilascio del certificato validamente firmato, l'identità dell'utente viene autenticata.
Si andrà infine a descrivere il protocollo SSL Handshake, tramite il quale si implementa ulteriormente la sicurezza delle connessioni fra client e server, creando un sistema sicuro di scambio dei parametri crittografici fra le due entità. Tuttavia si vedrà come anche questo protocollo sia vulnerabile dall'attacco MitM.

Indice

Capitolo 1

Crittografia

La crittografia è lo studio della codifica e della decodifica dei dati; tale scienza studia dei sistemi atti a rendere certe informazioni segrete e leggibili solo a chi possiede la chiave per decifrarle. Oggi è utilizzata quasi esclusivamente per sistemi di sicurezza, essendo in grado di fornire la maggior parte dei servizi contemplati nell'architettura di sicurezza stabilita sall'**ISO** (International Organization for Standardization). Pertanto i sistemi crittografici rivestono un'importanza fondamentale nella soluzione di molte problematiche di sicurezza relative alla protezione di informazioni.

1.1 L'algoritmo RSA

L'algoritmo RSA, a crittografia asimmetrica, fu inventato nel 1977 da Ronald Rivest, Adi Shamir e Leonard Adleman (da cui il nome RSA).
Tale sistema di crittografia si basa sull'esistenza di due chiavi distinte, la prima usata per cifrare il messaggio, e la seconda per decifrarlo. Sebbene le due chiavi siano dipendenti l'una dall'altra, è fondamentale non riuscire a risalire dall'una all'altra, garantendo così la segretezza e l'integrità del messaggio.

1.1.1 Generazione delle chiavi tramite RSA

1. L'utente A sceglie due numeri primi p e q diversi fra loro e molto grandi, dell'ordine di 1024 bit (circa 300 cifre), usando il Test di Miller-Rabin: si incrementa di 2 il numero dispari che fallisce il test, che viene poi

ripetuto fin quando la probabilità che il numero sia composto è minore di quella di un errore nei circuiti del computer;

2. si calcola $n = pq$, che diventa il **modulo** (di 2048 bit, ovvero circa 700 cifre);
3. si sceglie un valore dispari: e, tale che:
   - $1 < e < \varphi(n)$, dove $\varphi(n) = (p-1)(q-1)$;
   - $MCD(e, \varphi(n)) = 1$.

   Tale valore si dice **esponente di cifratura**;
4. l'utente A calcola $d = e^{-1} \bmod \varphi(n)$ usando l'identità di Bezout;
5. (n, e) è la chiave pubblica dell'utente A, mentre d è quella privata.

È importante che p e q, così come d, restino segreti.

1.1.2 Cifratura e decifrazione delle chiavi

- **Cifratura**: supponiamo che l'utente B voglia inviare un messaggio m all'utente A, la cui chiave pubblica è (n_A, e_A).
  Il messaggio m è tale che $0 < m < n_A$; se questo non fosse verificato, il messaggio viene diviso in blocchi. Per aumentare la sicurezza, si aggiungono a m dei bit casuali in modo che $\sqrt[e_A]{n_A} < m < n_A$, rendendo così necessaria la riduzione modulo n_A.
  Il messaggio cifrato diventa $c = m^{e_A} \bmod n_A$.
- **Decifrazione**: l'utente A riceve c, e lo eleva all'esponente segreto d_A. In questo modo A riesce a risalire al messaggio in chiaro.
  Il teorema seguente mostra come avviene questo ...processo....:

Teorema. *Si ha:*

$$(m^{e_a})^{d_a} \equiv m \mod n_a, \quad \forall\, m \in \{0, 1, \ldots, n_a - 1\}$$

Dimostrazione. Per costruzione, abbiamo:

$$ed = 1 + k\varphi(n) = 1 + k(p-1)(q-1), \quad \text{con} \quad k \in \mathbb{Z}$$

Quindi

$$(m^e)^d = m^{ed} = m^{1+k(p-1)(q-1)} = m(m^{p-1})^{q-1}$$

Ricordando che p è primo, distinguiamo due casi:

- se $MCD(m,p) = 1$, allora:

$$m^{p-1} \equiv 1 \mod p \implies m^{ed} \equiv m \cdot 1^{k(q-1)} \equiv m \mod p$$

- se $MCD(m,p) \neq 1$, allora:

$$p|m \implies m \equiv 0 \mod p \quad \text{e} \quad m \cdot m^{k(p-1)(q-1)} \equiv 0 \mod p$$

In ogni caso abbiamo $m \equiv m^{ed} \mod p$; analogamente si verifica che $m \equiv m^{ed} \mod q$.
Questo significa che $p|(m^{ed} - m)$ e $q|(m^{ed} - m)$.
Ora, poiché p e q sono primi, si ha $pq = n|(m^{ed} - m)$; quindi $m \equiv m^{ed} \mod n$. □

1.2 L'algoritmo Diffie-Hellman

L'algoritmo Diffie-Hellman nasce nel 1976, ed è uno dei più antichi algoritmi a chiave pubblica; esso è particolarmente adatto per la generazione di una chiave segreta tra due utenti che comunicano su un canale insicuro. L'efficienza dell'algoritmo sta nella complessità computazionale del logaritmo discreto.
Lo scopo dell'algoritmo Diffie-Hellman è quello di consentire a due utenti di scambiarsi dei parametri per creare una chiave segreta che verrà poi utilizzata per crittografare ulteriori messaggi. Nell'algoritmo Diffie-Hellman vi sono due parametri noti: un numero primo q e un intero α che è una radice primitiva modulo q.

1.2.1 Lo scambio delle chiavi con l'algoritmo Diffie-Hellman

Si supponga che gli utenti A e B vogliano scambiarsi una chiave:

1. l'utente A (o B) sceglie un numero primo p per il quale sia difficile il calcolo del logaritmo discreto, e una radice primitiva α mod p.
   I parametri α e p vengono resi pubblici;

2. l'utente A sceglie un numero segreto casuale X_A, con $1 \leq X_A \leq p-2$;

3. allo stesso modo l'utente B sceglie un altro numero segreto casuale X_B, tale che $1 \leq X_B \leq p-2$;

4. l'utente A invia a B il parametro $Y_A = \alpha^{X_A} \bmod p$;
5. l'utente B riceve Y_A e calcola $K = (Y_A)^{X_B} \bmod p$;
6. l'utente B invia ad A il parametro $Y_B = (\alpha)^{X_B} \bmod p$;
7. l'utente A riceve Y_B e calcola $K = (Y_B)^{X_A} \bmod p$.

In questo modo gli utenti A e B condividono la stessa chiave segreta K. Infatti:

$(Y_A)^{X_B} \bmod p = [(\alpha)^{X_A}]^{X_B} \bmod p = [(\alpha)^{X_B}]^{X_A} \bmod p = (Y_B)^{X_A} \bmod p$

Capitolo 2

La Sicurezza nel Web

2.1 Chiave pubblica e privata

Nella crittografia a chiave pubblica vengono utilizzate due chiavi per crittografare e decrittografare il messaggio: la chiave privata è nota solo al proprietario, mentre la chiave pubblica viene messa a disposizione delle altre entità presenti in rete.
La chiave pubblica, infatti, viene utilizzata dagli altri utenti per poter criptare il messaggio e inviarlo al destinatario; ques'ultimo, poi, procederà alla decifrazione del messaggio tramite la sua chiave privata. Il destinatario del messaggio, essendo l'unico a conoscere tale chiave, avrà la certezza che il messaggio sarà accessibile solo a lui.

2.2 Varie minacce e l'attacco Man in the Middle

Quando si usa il Web si va incontro a vari problemi di sicurezza. Questi consistono in attacchi attivi e passivi. Fra gli attacchi passivi vi sono l'intercettazione del traffico di rete fra browser e server e l'accesso a informazioni su un server Web che dovrebbe essere riservato. Invece, fra gli attacchi attivi vi sono la simulazione di altre identità, la modifica dei messaggi in transito fra client e server e l'alterazione delle informazioni contenute su un sito Web. Qui di seguito si riportano le minacce più frequenti:

- **minacce all'integrità**:
  - modifica dei dati utente;
  - cavalli di Troia;

  - modifiche della memoria;
  - modifiche dei messaggi in transito;

  Le conseguenze a questo tipo di minacce sono: perdita di informazioni, violazione della macchina, vulnerabilità ad altre minacce;

- **minacce alla segretezza**:

  - intercettazioni in rete;
  - furto di informazioni dal server;
  - furto di dati dal client;
  - informazioni sulla configurazione della rete;
  - informazioni sulla comunicazione fra client e server.

  Le conseguenze sono la perdita di informazioni e della privacy;

- **attacchi denial of service**:

  - terminazione dei processi degli utenti;
  - intasamento della macchina con una grande quantità di richieste;
  - saturazione dei dischi e della memoria;
  - isolamento della macchina tramite attacchi DNS.

  Questa minaccia impedisce all'utente di completare il proprio lavoro;

- **autenticazione**:

  - simulazione di utenti legittimi;
  - falsificazione dei dati.

  Tale minaccia porta ad un'errata rappresentazione dell'utente e all'accettazione di informazioni false ma ritenute valide.

2.2.1 L'attacco Man in the Middle

La tipologia di attacco che va sotto il nome di Man in the Middle (MitM) consiste nel dirottare il traffico generato durante la comunicazione fra due utenti verso un attaccante, che riuscirà così ad intercettare la comunicazione fra essi. Attraverso questo attacco l'hacker si inserisce fra le due entità, impersonando uno o entrambi gli utenti nei confronti dell'altro, accedendo in questo modo ad informazioni riservate.

A seconda della capacità di poter monitorare solo uno o entrambi i versi della connessione, l'attacco prende il nome di **Man in the Middle halfduplex** o **Man in the Middle full duplex**.
Per far sì che la sua presenza non sia percepita dalle vittime, l'hacker, una volta intercettato il messaggio, si dovrà preoccupare di inoltrarlo verso la corretta destinazione.
Un tipo comune di attacco MitM è quello in cui l'hacker usa una rete Wi-Fi: modifica la connessione del router, o sfrutta una debolezza nel setup del router per poter intercettare le sessioni degli utenti sul router. In questo caso l'hacker potrebbe configurare il dispositivo wireless della vittima, per esempio un portatile, impersonando un hotspot gratuito, dandogli un nome comunemente usato nelle aree Wi-Fi pubbliche. Successivamente, quando l'utente si connette al falso router e accede ai dati sensibili, come possono essere il sito della sua banca o pagine commerciali, l'hacker identifica la debolezza nella configurazione o nel sistema di crittografia del router e sfrutta tale debolezza per intercettare la comunicazione fra utente e router. In questo modo, l'hacker è in grado di spiare le sessioni che la vittima stabilisce con il router e rubare una grande quantità di dati. Bisogna però tener conto del fatto che i dati vengono scambiati in forma cifrata; occorre perciò poter violare questa cifratura. Come questo venga fatto comunemente, verrà spiegato in seguito.
L'attacco MitM può assumere diverse forme, fra cui una frequente è detta **ARP cache poisoning**.
Il protocollo ARP (Address Resolution Protocol) ha il compito di mantenere in una cache sul PC tutte le associazioni IP-MAC Address di tutti i computer raggiungibili dal PC che si trovano sullo stesso segmento di rete locale. Normalmente, nella comunicazione tra due utenti A e B si ha una situazione del tipo descritto nel seguente esempio:

UTENTE A	**UTENTE B**
192.168.1.50	192.168.1.100
MAC: 00:00:00:aa:aa:aa	MAC: 00:00:00:bb:bb:bb
ARP cache	ARP cache
192.168.1.100 at 00:00:00:bb:bb:bb	192.168.1.50 at 00:00:00:aa:aa:aa

Nella ARP cache dell'utente A vi è la giusta associazione IP-MAC Address dell'utente B,e la comunicazione avviene senza problemi.
Se invece si ha un caso di ARP cache poisoning:

UTENTE A	**UTENTE B**
192.168.1.50	192.168.1.100
MAC: 00:00:00:aa:aa:aa	MAC: 00:00:00:bb:bb:bb
ARP cache poisoned	ARP cache poisoned
192.168.1.100 at 00:00:00:cc:cc:cc	192.168.1.50 at 00:00:00:cc:cc:cc

UTENTE C

192.168.1.150

MAC: 00:00:00:cc:cc:cc

ARP cache

192.168.1.100 at 00:00:00:bb:bb:bb

192.168.1.50 at 00:00:00:aa:aa:aa

Si osserva che l'utente A ha associato l'IP dell'utente B con il MAC Address dell'utente C, e analogamente B ha associato l'indirizzo IP di A al MAC Address di C. In questo modo ogni messaggio inviato da A o B conterrà un MAC Address sbagliato e raggiungerà l'utente C. Quest'ultimo può intercettare i pacchetti che transitano sulla sua scheda di rete ed eventualmente inoltrarli ai corretti destinatari.

Di recente è nata una variante dell'attacco MitM, chiamata **Man in the Browser**. In questo caso, l'hacker riesce ad installare un codice malware nel

computer della vittima, che agisce all'interno del browser. Il malware è in grado di registrare i dati scambiati fra il browser e i siti con cui si connette il browser.
Questo tipo di attacco si è diffuso in larga scala negli ultimi anni, poiché permette all'hacker di attaccare contemporaneamente più utenti e l'attacco può essere effettuato in modalità remota. Ci sono vari metodi per difendersi dagli attacchi MitM, anche se quasi tutti gli attacchi sono diretti al router e ai server e questo impedisce agli utenti di controllare in prima persona la sicurezza delle transazioni. Tuttavia si può usare un programma di crittografia efficace che agisca fra client e server. Grazie a programmi di questo tipo, il server può autenticarsi presentando un certificato digitale; così client e server possono stabilire un canale criptato attraverso il quale inviare dati sensibili. Il problema nasce proprio con la necessità che il certificato digitale sia autentico.

2.2.2 Un esempio di attacco MitM: l'algoritmo Diffie-Hellman

Il protocollo per lo scambio delle chiavi Diffie-Hellman è vulnerabile all'attacco MitM, che viene così organizzato:

1. l'utente A (o B) sceglie un numero primo p per il quale sia difficile il calcolo del logaritmo discreto, e una radice primitiva α mod p.
   I parametri α e p vengono resi pubblici;

2. l'utente A sceglie un numero segreto casuale X_A, con $1 \leq X_A \leq p-2$;

3. allo stesso modo l'utente B sceglie un altro numero segreto casuale X_B, tale che $1 \leq X_B \leq p-2$;

4. con lo scopo di intromettersi nella comunicazione fra A e B, l'utente C genera casualmente due parametri X_{C_1} e X_{C_2}, calcolando poi i parametri pubblici:

   - $Y_{C_1} = \alpha^{X_{C_1}} \bmod p$;
   - $Y_{C_2} = \alpha^{X_{C_2}} \bmod p$;

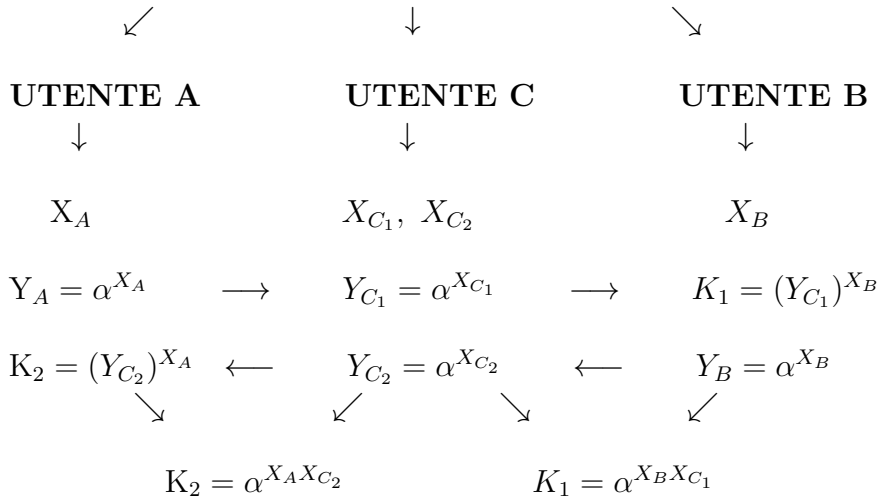
5. l'utente A invia all'utente B il parametro $Y_A = \alpha^{X_A} \bmod p$;

6. l'utente C intercetta Y_A e trasmette Y_{C_1} a B, fingendosi A;

7. intanto C genera $K_2 = (Y_A)^{X_{C_2}} \bmod p$, che sarà la chiave segreta condivisa con l'utente A, impersonando B;

8. l'utente B riceve Y_{C_1} e calcola $K_1 = (Y_{C_1})^{X_B} \bmod p$;
9. l'utente B trasmette all'utente A il numero $Y_B = \alpha^{X_B} \bmod p$;
10. l'utente C intercetta Y_B e trasmette ad A il parametro Y_{C_2}, spacciandosi per B;
11. inoltre C calcola $K_1 = (Y_B)^{X_{C_1}} \bmod p$, che sarà la chiave segreta condivisa con B;
12. l'utente A riceve Y_{C_2} e calcola $K_2 = (Y_{C_2})^{X_A} \bmod p$.

Parametri pubblici: α, p



Ora i due utenti A e B pensano di condividere una chiave segreta, ma in realtà B e C condividono la chiave segreta K_1, mentre A e C condividono la chiave segreta K_2.
Da questo momento, tutte le comunicazioni successive fra A e B verranno compromesse nel modo seguente:

1. A invia a B il messaggio crittografato con la chiave K_2 (che crede di condividere con B);
2. C intercetta il messaggio crittografato e lo decifra tramite K_2;

3. C cifra di nuovo il messaggio tramite K_1 e lo invia a B. In questo modo C si limita a spiare la comunicazione fra A e B senza alterarla. Altrimenti C potrebbe anche modificare il messaggio destinato a B, o anche generare nuovi messaggi;

4. Quando B invia un messaggio ad A crittografato con la chiave K_1, l'utente C lo intercetta, lo decripta, lo cifra di nuovo con la chiave K_2 e lo invia ad A.

L'attacco al protocollo Diffie-Hellman può essere evitato con l'uso delle firme digitali e dei certificati a chiave pubblica.

2.3 Infrastruttura a chiave pubblica e Autorità di certificazione

Un'**infrastruttura a chiave pubblica** (PKI) rappresenta un insieme di direttive, di politiche e risorse umane che definiscono le procedure per la generazione e la pubblicazione di chiavi crittografiche legate ad un'entità, che sia un utente o un'informazione.
Questa associazione chiave-utente viene resa valida tramite l'emissione di un certificato digitale, che presenta un insieme di informazioni firmate e rilasciate da un'autorità fidata, la **Certification Authority**.
Vi sono vari tipi di certificati; tra i due più noti si ricordano:

- i certificati di identità, che contengono informazioni sull'identità di un'entità (per esempio l'indirizzo di posta elettronica) e la chiave pubblica dell'entità;

- il certificato credenziale, che contiene informazioni relative ai diritti di accesso a una risorsa.

Il certificato contiene la chiave pubblica dell'utente, e viene firmato tramite la chiave privata della CA. Senza la firma digitale dell'autorità di certificazione, non fornisce di per sé autenticazione.
Esso viene pubblicato sul sito Internet dell'autorità stessa, e viene spesso incluso nei browser abilitati alle tecnologie PKI, per favorirne la diffusione: Explorer, Netscape/Mozilla ed Opera vengono distribuiti con alcuni certificati root già installati, così che l'utente ha a disposizione una lista di CA risconosciute dal proprio browser, e gli viene anche data la possibilità di installare nuovi certificati, in modo che questi ultimi vengano riconosciuti automaticamente dal programma.

Se invece il browser incontra un certificato firmato da una CA non riconosciuta, l'evento viene segnalato all'utente, che potrà decidere se rigettare il certificato, oppure accettarlo, ed inserire la nuova CA fra quelle riconosciute, perché abbia accesso alla sua chiave pubblica autentica.
Si presuppone che il certificato firmato dalla CA sia affidabile, data la buona reputazione che essa è tenuta a mantenere, sotto compenso economico.

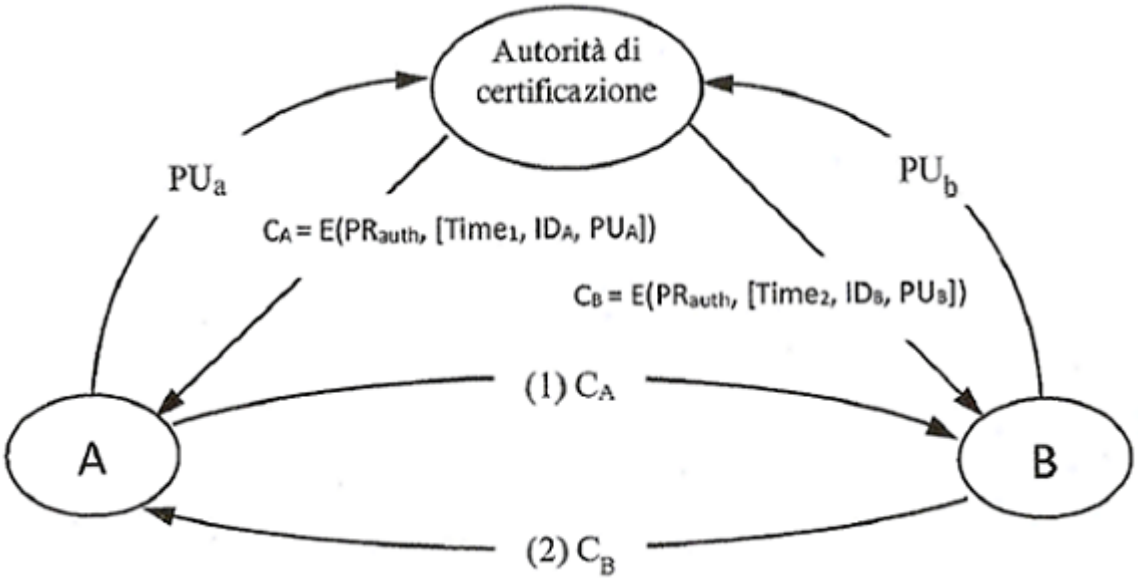


Figura 2.1: Scambio di certificati a chiave pubblica.

La figura mostra la CA che invia all'utente A il certificato nella forma: $C_A = E(PR_{auth}, [Time_1, ID_A, PU_A])$, dove:

- PR_{auth} è la chiave privata della CA;
- $Time_1$ è un timestamp;
- ID_A è l'identità dell'utente A;
- PU_A è la chiave pubblica di A.

Analogamente l'utente B riceve dalla CA il certificato nella forma:
$C_B = E(PR_{auth}, [Time_2,], ID_B, PU_B)$.

2.3.1 Gerarchia di certificazione

Per ragioni di efficienza, spesso la CA delega la firma dei certificati a una o più **autorità di registrazione** (RA).
Ogni RA ha un proprio certificato, firmato dalla CA. L'autorità di registrazione effettua l'identificazione degli utenti che richiedono il rilascio di un certificato e fornisce gli strumenti necessari per la creazione delle chiavi.

2.3.2 Requisiti di sicurezza

Tramite l'utilizzo del certificato digitale e della chiave pubblica in esso contenuta, la PKI consente di soddisfare vari requisiti:

- **Autenticazione**: garantisce l'identità del titolare del certificato, ovvero del possessore della chiave privata relativa alla chiave pubblica contenuta nel certificato;
- **Confidenzialità**: garantisce che solo il destinatario del messaggio possa accedere al contenuto;
- **Integrità**: garantisce che l'informazione non sia stata alterata durante la trasmissione, sia da parte di utenti e processi non autorizzati, sia per eventi accidentali;
- **Non ripudio**: garantisce che il mittente del messaggio non possa negare di averlo spedito, e che il destinatario non possa negare di averlo ricevuto.

2.3.3 Autenticazione del certificato

L'autenticazione del certificato è fondamentale per la sicurezza delle comunicazioni: l'utente A deve essere in grado di provare la sua identità all'utente B, e allo stesso tempo deve poter verificare l'identità del suo interlocutore.
Se questo non fosse possibile, potrebbe verificarsi l'intercettazione del messaggio da parte di un intruso, che sarà così in grado di assumere l'identità di uno dei due utenti, o addirittura di entrambi, come si è visto nell'attacco MitM al protocollo Diffie-Hellman.
Per poter verificare l'autenticità del messaggio, è necessario aver già testato la legittimità del certificato; occorre quindi procurarsi il certificato della CA, estrarne la chiave pubblica ed utilizzarla per testare la veridicità della firma presente sul certificato del messaggio ricevuto. Per poter verificare la validità del certificato, inoltre, occorre controllare la data di scadenza e verificare presso la base di dati della CA che tale certificato non sia presente nelle liste

di revoca.
Se siamo in presenza di un certificato emesso da una gerarchia di CA, è necessario risalire al certificato della Root CA per poter controllare a catena la legittimità dei certificati delle CA subordinate.

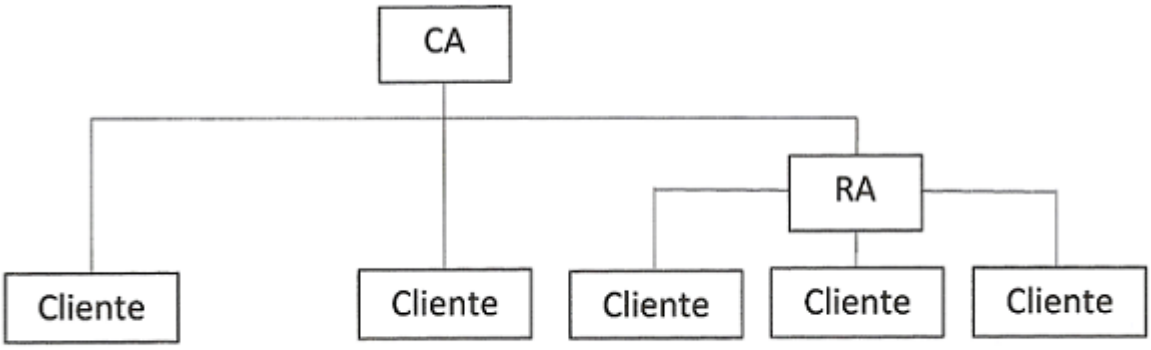


Figura 2.2: Un esempio di gerarchia di certificazione.

2.3.4 Il certificato X.509

Il tipo di certificato maggiormente utilizzato segue lo standard X.509, che fu progettato dall'**Internet Engineering Task Force** e presentato per la prima volta nel 1988.
Il certificato X.509 include varie voci:

1. **Gerarchia certificato**: elenca la catena di CA che hanno emesso il certificato;

2. **Versione**: indica la versione del certificato; ve ne sono tre: la prima è stata emessa nel 1988, mentre la più recente risale al 1997;

3. **Numero seriale**: ogni certificato rilasciato dalla CA è contraddistinto da un numero seriale;

4. **Algoritmo di firma del certificato**: è l'algoritmo con cui è stata prodotta la firma del certificato;

5. **Emittente**: è il nome della CA che rilascia il certificato; in questa voce troviamo sigle quali:

- **OU**: l'unità entro l'organizzazione;
- **O**: l'organizzazione;
- **C**: il paese in cui è stato emesso il certificato;

6. **Validità**: in questa voce è annotato il periodo di validità del certificato: la data della sua emissione e della scadenza;

7. **Soggetto**: Il nome del titolare del certificato;

8. **Chiave pubblica soggetto**: essa può essere espressa mediante cifratura RSA o Diffie-Hellman;

9. **Valore firma certificato**: è la firma digitale dell'autorità di certificazione, che attesta la validità del legame tra la chiave pubblica e le informazioni di identificazione del soggetto.
   Per produrre la firma digitale, si calcola l'hash delle informazioni di questo certificato usando l'algoritmo specificato, e lo si firma usando la chiave privata della CA.

2.3.5 Utilizzo del certificato digitale

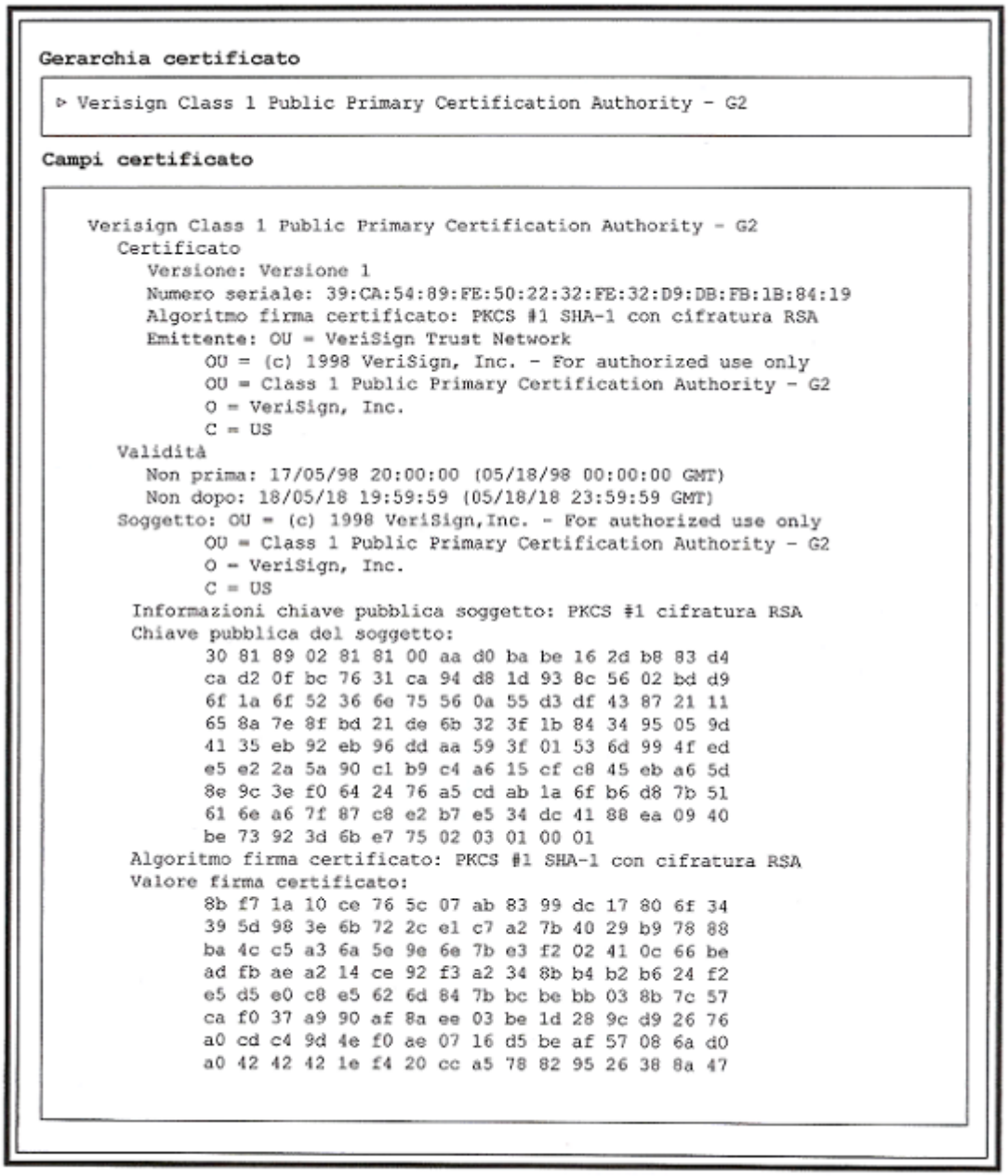
Esempi di utilizzo del certificato digitale:

1. **certificato SSL**, tramite cui si stabilisce la comunicazione client-server sul Web: il client, per autenticarsi, invia il proprio certificato al server che risponde a sua volta inviando al client il proprio certificato, in modo da instaurare una comunicazione sicura. Dopo essersi autenticati, client e server si scambiano una chiave segreta di sessione e possono poi scambiarsi messaggi e informazioni criptati;

2. **certificati S/MIME**, utilizzati per firmare e criptare e-mail: solo la firma digitale, infatti, assicura l'identità del mittente e l'integrità del messaggio;

3. **certificato object signing**, utilizzato per firmare codici come Java, Javascript, o altri tipi di file; ciò permette alla ditta produttrice di software di garantire ai suoi clienti l'autenticità del prodotto;

4. **commercio elettronico**: si utilizza il certificato per transazioni commerciali, assicurando ai due contraenti di potersi autenticare l'un l'altro in modo inequivocabile.



Gerarchia certificato

▷ Verisign Class 1 Public Primary Certification Authority - G2

Campi certificato

```
Verisign Class 1 Public Primary Certification Authority - G2
   Certificato
      Versione: Versione 1
      Numero seriale: 39:CA:54:89:FE:50:22:32:FE:32:D9:DB:FB:1B:84:19
      Algoritmo firma certificato: PKCS #1 SHA-1 con cifratura RSA
      Emittente: OU = VeriSign Trust Network
            OU = (c) 1998 VeriSign, Inc. - For authorized use only
            OU = Class 1 Public Primary Certification Authority - G2
            O = VeriSign, Inc.
            C = US
   Validità
      Non prima: 17/05/98 20:00:00 (05/18/98 00:00:00 GMT)
      Non dopo: 18/05/18 19:59:59 (05/18/18 23:59:59 GMT)
   Soggetto: OU = (c) 1998 VeriSign,Inc. - For authorized use only
            OU = Class 1 Public Primary Certification Authority - G2
            O = VeriSign, Inc.
            C = US
    Informazioni chiave pubblica soggetto: PKCS #1 cifratura RSA
    Chiave pubblica del soggetto:
            30 81 89 02 81 81 00 aa d0 ba be 16 2d b8 83 d4
            ca d2 0f bc 76 31 ca 94 d8 1d 93 8c 56 02 bd d9
            6f 1a 6f 52 36 6e 75 56 0a 55 d3 df 43 87 21 11
            65 8a 7e 8f bd 21 de 6b 32 3f 1b 84 34 95 05 9d
            41 35 eb 92 eb 96 dd aa 59 3f 01 53 6d 99 4f ed
            e5 e2 2a 5a 90 c1 b9 c4 a6 15 cf c8 45 eb a6 5d
            8e 9c 3e f0 64 24 76 a5 cd ab 1a 6f b6 d8 7b 51
            61 6e a6 7f 87 c8 e2 b7 e5 34 dc 41 88 ea 09 40
            be 73 92 3d 6b e7 75 02 03 01 00 01
    Algoritmo firma certificato: PKCS #1 SHA-1 con cifratura RSA
    Valore firma certificato:
            8b f7 1a 10 ce 76 5c 07 ab 83 99 dc 17 80 6f 34
            39 5d 98 3e 6b 72 2c e1 c7 a2 7b 40 29 b9 78 88
            ba 4c c5 a3 6a 5e 9e 6e 7b e3 f2 02 41 0c 66 be
            ad fb ae a2 14 ce 92 f3 a2 34 8b b4 b2 b6 24 f2
            e5 d5 e0 c8 e5 62 6d 84 7b bc be bb 03 8b 7c 57
            ca f0 37 a9 90 af 8a ee 03 be 1d 28 9c d9 26 76
            a0 cd c4 9d 4e f0 ae 07 16 d5 be af 57 08 6a d0
            a0 42 42 42 1e f4 20 cc a5 78 82 95 26 38 8a 47
```

Figura 2.3: Un esempio di certificato digitale. Si osservino le ultime tre coppie di cifre presenti nella chave pubblica del soggetto (10 00 01): corrispondono all'esponente di cifratura che equivale a $2^{16} + 1$. Le cifre precedenti sono il modulo.

2.4 Attacchi effettuati negli ultimi anni

1. **Stuxnet**: nel giugno 2010 viene identificato un nuovo **warm**, diffuso tramite una chiavetta USB da cui vengono installati device driver validamente firmati da Verisign. Il 60% dei sistemi colpiti si trova in Iran, e si ritene che questo sia stato un attacco per sabotare i sistemi degli impianti di arricchimento dell'uranio per le centrali nucleari iraniane. I certificati appartengono a due aziende elettroniche taiwanesi, che hanno sede in edifici adiacenti. Questo fa pensare ad un accesso fisico ad entrambe le sedi, ma non è ancora chiaro come possa essere accaduto. Verisign ha così revocato i certificati, ed è stato implementato un aggiornamento Windows per distribuire la revoca in tutto il mondo.

2. Nel 2011 la CA **Comodo**, a causa della compromissione di una RA subordinata, ha firmato nove certificati fraudolenti, intestati a:
   mail.google.com, www.google.com, login.yahoo.com (tre certificati), login.skype.com, addons.mozilla.com.
   Comodo ha poi revocato i certificati, affermando che l'attacco ha compromesso la sola RA, ma era stato pianificato da tempo e proveniva da un IP collocato in Iran.

3. Nel settembre 2011, la CA olandese **DigiNotar** fu liquidata dopo la scoperta di un'ampia compromissione dei sistemi.
   Fu grazie a uno studente iraniano che si scoprì un certificato fraudolento emesso da DigiNotar, a causa di un'intrusione nella CA. Nei giorni successivi, si scopriranno molti altri certificati che DigiNotar ha dovuto revocare. Il 2 settembre 2011 il Governo olandese ritira la fiducia a DigiNotar, con un conseguente aggiornamento di Windows e degli altri browser, che hanno eliminato DigiNotar dalla lista delle CA riconosciute. I certificati emessi fraudolentemente in questi anni sono più di 531, intestati ad aziende quali Android, Google, Microsoft, Mozilla, Skype, Twitter, Facebook, Yahoo, Torproject, oltre che ad altre CA e addirittura ad enti governativi: CIA, Mossad, MI5. Anche in questo caso, le richieste di uso dei certificati hanno rivelato che l'area del loro utilizzo era concentrata soprattutto in Iran; ciò porta a ritenere che sia stata un'azione governativa con lo scopo di generare un sistema 'man in the middle' per l'intercettazione sistematica della posta scambiata su Gmail.

4. **Flame**:il 28 maggio 2012 viene scoperto un nuovo 'attack toolkit': si tratta di un complesso e sofisticato sistema di spionaggio in grado di catturare selettivamente file, immagini, conversazioni audio. Esso fu

scoperto a seguito di un'indagine su un virus nel ministero del petrolio in Iran.
Flame ha ottime capacità stealth, ovvero si automodifica in funzione della presenza di antivirus, e installa driver il cui codice è regolarmente firmato da un certificato emesso da una CA Microsoft. Quest'ultima, il 3 giugno 2012, ha emesso un advisory per correggere il problema ed ha revocato due CA intermedie e i relativi certificati: la Microsoft Enforced Licensing Intermediate PCA e la Microsoft Enforced Licensing Registration Authority.

Questi episodi mostrano come spesso la PKI,e la relativa catena di certificazione, siano l'anello debole della sicurezza su Internet. Il problema dell'integrità della catena di fiducia è ancora sottovalutato: quasi nessun browser accede alle liste di revoca per verificarne la validità dei certificati e molte CA rilasciano certificati senza troppi controlli.
Vediamo ora in dettaglio un protocollo di uso universale: SSL (o TLS, la sua versione aggiornata), per la cui sicurezza la PKI gioca un ruolo cruciale.

2.5 Il protocollo SSL

Il protocollo SSL, **Secure Socket Layer**, fu progettato da Netscape e viene utilizzato per stabilire comunicazioni sicure fra client e server sul Web, assicurando protezione da intrusioni, manomissioni e falsificazioni dei messaggi.
Questo viene reso possibile tramite tre funzionalità:

1. Sicurezza del collegamento: i dati trasmessi da entrambi gli utenti coinvolti nella comunicazione vengono criptati tramite algoritmi crittografici a chiave simmetrica;

2. Autenticazione: l'identità del client e del server viene garantita tramite i rispettivi certificati e lo scambio delle chiavi crittografiche, attraverso uno schema challenge_response;

3. Affidabilità: il messaggio viene protetto e mantenuto integro grazie al MAC, **Message Authentication Code**: è un messaggio contenente una funzione hash a chiave segreta che assicura che la trasmissione non sia stata alterata, e garantire l'origine del messaggio.

Al fine di comprendere il funzionamento del protocollo SSL, è necessario specificare il significato di sessione e connessione:

- la **connessione** è un'associazione a livello di trasporto che fornisce un certo tipo di servizio, ad esempio un trasporto affidabile di dati. Le connessioni sono utilizzate all'interno di una sessione SSL, che è una associazione fra client e server, stabilita mediante il protocollo di handshake, e nella quale sono definiti i parametri crittografici usati nella comunicazione.

- la **sessione SSL** è un'associazione fra un client e un server. Essa definisce un insieme di parametri di sicurezza crittografica creati dal protocollo SSL Handshake, necessari per l'instaurazione di una comunicazione sicura;

Il protocollo SSL è composto da più parti:

- **SSL Record**: questo protocollo provvede alla cifratura del messaggio: i dati vengono frammentati in blocchi di 2^{14} byte o meno, ed eventualmente compressi; viene poi calcolato il MAC tramite la chiave segreta e si procede così alla crittografia simmetrica dei dati e del MAC. Si aggiunge infine l'intestazione contenente le voci relative al tipo di conenuto;

- **Protocollo Handshake**: è la parte più complessa di SSL, e permette di creare una connessione sicura fra client e server. Il protocollo verrà mostrato nel dettaglio nella sezione seguente.

2.5.1 Il protocollo SSL Handshake e lo scambio delle chiavi

Il protocollo Handshake permette al client e al server di autenticarsi l'un l'altro e di accorddarsi un algoritmo di crittografia per la cifratura e decifrazione dei messaggi, il MAC e le chiavi crittografiche da utilizzare.
Il protocollo Handshake è caratterizzato da un insieme di messaggi scambiati fra client e server, suddivisi in quattro fasi, al termine delle quali sarà garantita una connessione sicura:

1. **Inizializzazione funzionalità di sicurezza**: il client invia al server un messaggio chiamato 'client_hello', il cui contenuto presenta vari campi:

   - **Random**: è una sequenza di numeri casuali generata dal client e costituita da un timestamp di 32 bit e di 28 byte, prodotti da un generatore sicuro di numeri casuali; questa sequenza funge da codice **nonce** e protegge da eventuali attacchi a **replay** durante lo scambio delle chiavi, mediante il valore timestamp;

- **Session ID**: è un identificatore di sessione di varia lunghezza. Se presenta un valore diverso da zero, significa che il client vuole aggiornare una connessione già esistente, o creare una nuova connessione nella stessa sessione. Se invece troviamo il valore zero, il client desidera aprire una connessione da una nuova sessione;
- **CipherSuite**: rappresenta una lista di algoritmi crittografici supportati dal client;
- **Compression Method**: rappresenta un elenco dei metodi di compressione supportati dal client.

Dopo aver ricevuto il messaggio 'client_hello', il server risponde con il 'server_hello', in cui vengono presentate le stesse voci:

- **Random**: è una sequenza casuale di numeri generata dal server;
- **Version**: rappresenta la più bassa versione supportata dal client e la più alta supportata dal server;
- **Session ID**: dipende dal valore presente nella Session ID inviata dal client: se il valore è diverso da zero, il server usa lo stesso valore, altrimenti genera un valore per la nuova sessione;
- **CipherSuite**: contiene l'algoritmo crittografico per lo scambio delle chiavi e del MAC, scelto dal server tra quelli proposti dal client. I tipi di algoritmi maggiormente utilizzati sono:
  - **RSA**: la chiave segreta viene crittografata con la chiave pubblica RSA del destinatario;
  - **Fixed Diffie-Hellman**: il client e il server si scambiano i propri certificati (autenticati dalla firma digitale delle rispettive CA) contenenti i parametri pubblici Diffie-Hellman: la radice, il generatore e i parametri a e b. Da esse si calcola la chiave segreta fra i due nodi.
  - **Ephemeral Diffie-Hellman**: è l'algoritmo più sicuro: viene utilizzato per creare chiavi autenticate temporanee, monouso. In questa procedura vengono scambiati i parametri pubblici Diffie-Hellman, firmati tramite chiave privata RSA o DSS. Il destinatario verifica la firma con la corrispondente chiave pubblica.
  - **Anonymous Diffie-Hellman**: anche qui vengono scambiati i parametri pubblici Diffie-Hellman, ma senza autenticazione.
- **Compression Method**: contiene il metodo di compressione scelto dal server fra quelli supportati dal client.

2. **Autenticazione del server e scambio delle chiavi**: il server invia al client il messaggio **certificate**, in cui presenta il suo certificato X.509, o la catena dei certificati, se autenticato da una gerarchia di CA. Questo messaggio non è necessario se lo scambio delle chiavi viene effettuato tramite Anonymous Diffie-Hellman.
   Segue poi il messaggio **server_key_exchange**, in cui il server invia le informazioni per lo scambio delle chiavi.
   Tale messaggio viene inviato nei seguenti casi:

   - **Anonymous Diffie-Hellman**: all'interno del messaggio si trovano due valori globali Diffie-Hellman, ovvero un numero primo e una sua radice primitiva e la chiave Diffie-Hellman pubblica del server;
   - **Ephemeral Diffie-Hellman**: in questo caso, il messaggio server_key_exchange contiene gli stessi parametri utilizzati per l'Anonymous Diffie-Hellman, con la firma di questi parametri;
   - **Scambio delle chiavi RSA in cui il server ha una chiave RSA di sola firma**: in questo caso, il server crea una coppia di chiavi RSA pubblica e privata, e invia la chiave pubblica tramite il server_key_exchange. Nel messaggio saranno presenti i due parametri per la cifratura RSA: esponente e modulo, e la firma di tali parametri.

   Vi sono casi in cui il messaggio server_key_exchange non è necessario: quando si utilizza il Fixed Diffie-Hellman, visto che i parametri per lo scambio delle chiavi sono già presenti nel certificato inviato precedentemente dal server, e con lo scambio delle chiavi tramite cifratura RSA.
   A seguire, il server invia il messaggio **certificate_request**, in cui troviamo il **certificate_type**, ovvero il tipo di algoritmo a chiave pubblica e il suo impiego, e il **certificate_authorities**, un elenco di CA accettate dal server.
   La seconda fase si conclude con il messaggio obbligatorio del **server_done**, che non ha alcun parametro, ma serve solo a chiudere la lista di messaggi inviati dal server.

3. **Autenticazione del client e scambio delle chiavi**: in questa fase il client controlla che il certificato inviato dal server sia valido verificando la firma digitale della CA sul certificato; questo richiede la chiave pubblica della CA. Il client verifica anche che i parametri di server_hello siano accettabili. Se poi il server ha richiesto il certificato del client, quest'ultimo invia il messaggio **certificate**.

Si passa ora alla fase dello scambio delle chiavi con il messaggio **client_key_exchange** inviato dal client.
Come nella fase precedente, il contenuto del messaggio dipende dal tipo di scambio delle schiavi:

- **RSA**: il client genera un valore pre-master di 48 byte e lo cifra tramite la chiave pubblica del server presente nel suo certificato o la chiave RSA scambiata tramite il messaggio server_key_exchange;
- **Ephemeral o Anonymous Diffie-Hellman**: il client invia i parametri pubblici Diffie-Hellman;
- **Fixed Diffie-Hellman**: il client ha già provveduto ad inviare i parametri pubblici Diffie-Hellman tramite il messaggio certificate; in questo caso, quindi, il contenuto di client_key_exchange è nullo;

Successivamente il client invia il messaggio **certificate_verify**, in cui verifica esplicitamente il proprio certificato. Lo scopo del messaggio è quello di mostrare la proprietà della chiave privata da parte del client; in questo modo, anche se il certificato del client possa essere utilizzato fraudolentemente, non si riuscirebbe comunque a inviare il certificate_verify, non essendo in possesso della chiave privata del client.

4. **Conclusione**: Il protocollo SSL Hanshake si conclude con il messaggio **change_cipher_spec**, inviato dal client, che copia il CipherSpec temporaneo nel CipherSpec corrente. Segue il messaggio **finished** che contiene i nuovi algoritmi e le chiavi. Questo messaggio verifica che lo scambio delle chiavi e l'autenticazione siano avvenuti correttamente. Allo stesso modo il server invia il suoi messaggi **change_cipher_spec** e **finished**.

Qui di seguito si riporta uno schema che riassume tutti i passaggi del protocollo SSL Handshake.

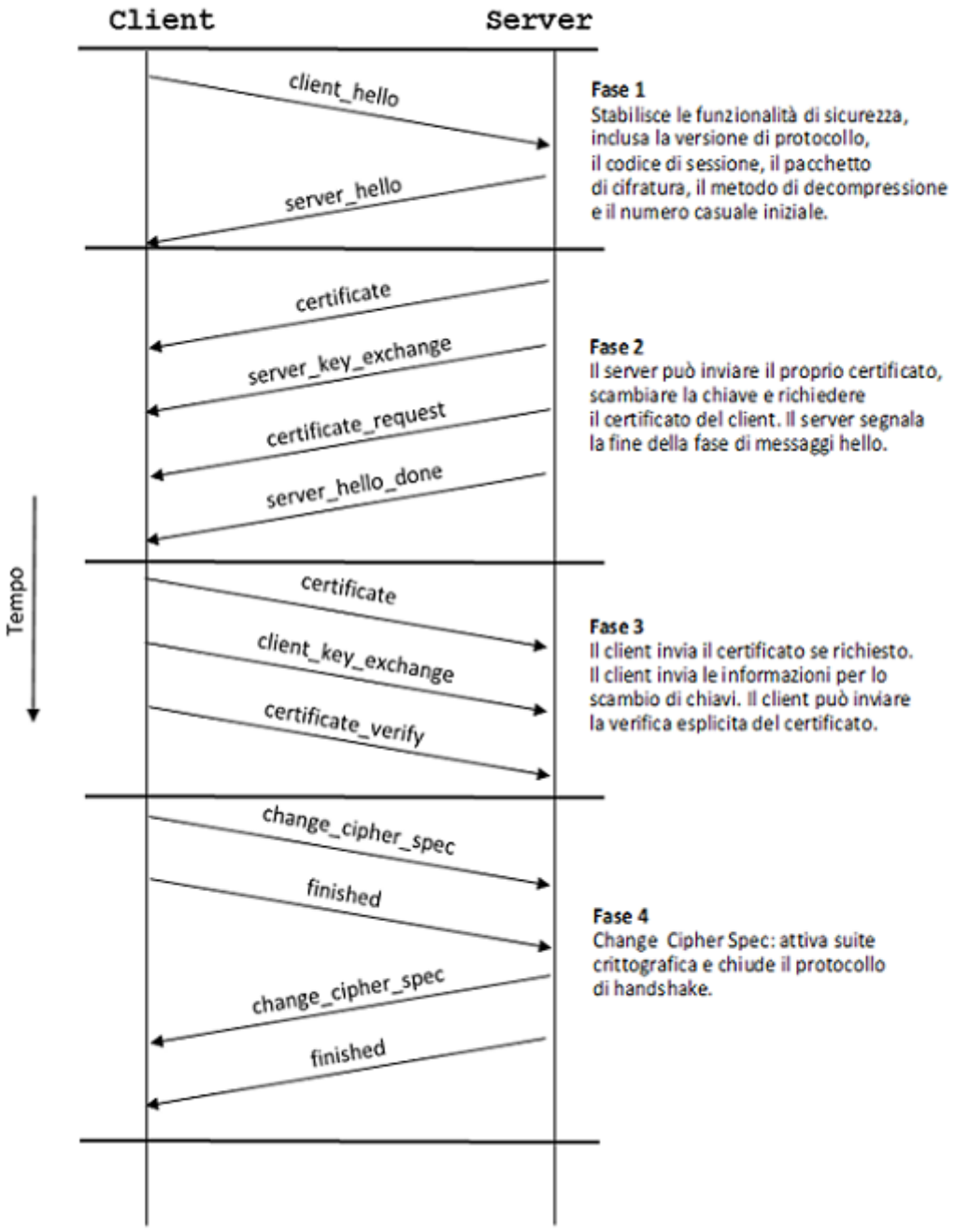


Figura 2.4: Il protocollo SSL Handshake. L'autenticazione del server è obbligatoria, quella del client solo facoltativa.

2.5.2 Violazione del protocollo SSL

Quando ci si collega ad un sito Web che gestisce informazioni sensibili, la connessione avviene utilizzando il protocollo **HTTPS** (HyperText Transfer Protocol over Secure Socket Layer), ovvero il classico HTTP con l'aggiunta del protocollo SSL/TLS. Tra client e server viene così stabilito un canale di comunicazione cifrato attraverso lo scambio di certificati digitali. Questi ultimi vengono utilizzati per attestare l'identità di un sito Web: il browser, non appena si collega al server Web remoto, può così stabilire se il sito è autentico o meno, e se corrisponde a quello che effettivamente dichiara di essere.
Quando il certificato risulta firmato da una CA riconosciuta, il browser dà il via al meccanismo di cifratura per scambiare i dati fra client e server in modo sicuro.
Ma tutto questo non è sufficiente per poter evitare eventuali attacchi MitM: se un hacker riuscisse in qualche modo a risalire ad un certificato valido, potrebbe far credere alla vittima di essere il server legittimo:

1. la vittima invia una richiesta di handshake all'hacker, pensando di essersi messo in comunicazione con il server;

2. l'hacker risponde con una chiave valida per se stesso;

3. il client cifra tramite la chiave pubblica dell'hacker il proprio messaggio di risposta con allegata la propria chiave;

4. in questo modo l'hacker potrà decriptare la richiesta e avrà la chiave per creare risposte valide per la vittima;

5. analogamente l'hacker stabilisce una seconda connessione cifrata col server legittimo, fingendosi il client; cifra nuovamente la richiesta del client con la chiave pubblica del server, e gliela invia;

6. quando il server invia la risposta all'hacker, quest'ultimo reinvia tale risposta alla vittima che non noterà nulla di strano.

Vi sono poi casi in cui il protocollo SSL può essere violato in modo pressoché legittimo: gli antivirus ne sono un esempio. Quando si installa un programma antivirus, si accettano dei termini e condizioni che permettono al programma di accedere ai propri dati personali; l'antivirus presenta inoltre un certificato root, considerato quindi valido, perché autofirmato. In questo modo l'antivirus si comporta a tutti gli effetti come un vero e proprio hacker, riuscendo ad effettuare un 'attacco' MitM: nel momento in cui il client tenta di aprire una sessione con il server, l'antivirus, per poter controllare la

legittimità del sito che il client vuole visitare, intercetta la connessione e si interpone fra client e server: impersonando il server stesso, legge la richiesta di handshake inviata dal client, scambiando così una chiave di sessione con il client. Allo stesso modo riuscirà scambiare un'altra chiave di sessione con il server, fingendo di essere il client.
Un caso simile si verifica all'interno delle aziende e luoghi di lavoro, in cui il datore di lavoro controlla i siti a cui i dipendenti si connettono: essi accettano di installare un certificato che permette all'azienda di legittimare tale attività di controllo.

Bibliografia

- [1] [Stallings 2007] W. Stallings, Crittografia e sicurezza delle reti, seconda edizione, McGraw Hill Education, 2007

- [2] [Trappe, Washington 2009] W. Trappe, L.C. Washington, Crittografia con elementi di teoria dei codici, Pearson, Prentice Hall, 2009

- [3] [Aliffi 2009] Aliffi Davide, Algoritmi di Teoria dei Numeri e Crittografia, note del corso, Università di Bologna, A.A 2008/2009

- [4] https://siamogeek.com/2013/08/https-come-funziona/

- [5] http://www.punto-bit.com/cgi-bin/main.pl?file=6la=it

- [6] http://home.deib.polimi.it/serazzi/sicurezza/materiale/FirmaDigitalePKI0506.pdf

- [7] https://certificatossl.wordpress.com/2013/01/03/ca-root-certificate-certificato-ssl-intermedio-certificato-radice/

- [8] http://www.jus.unitn.it/presidio/documentoelettronico/firmadigitale/Manuali/Info_FirmaDigitale.pdf

□ [9] http://www.hackerstribe.com/vocabolario/hijacking/mitm-man-in-the-middle/

□ [10] http://home.deib.polimi.it/serazzi/sicurezza/materiale/FirmaDigitalePKI0506.pdf

□ [11] http://www.guidami.info/2014/05/gli-antivirus-tracciano-attivita-utenti-inviano-file-personali.html

□ [12] https://www.securitysummit.it/archivio/2012/roma/upload/file/atti%20roma%202012/07.06.2012_GIUSTOZZI%20CORRADO.pdf

□ [13] https://it.wikipedia.org/wiki/X.509

□ [14] ftp://ftp.disi.unige.it/person/ChiolaG/sic03-04/s0312094.pdf

□ [15] http://security.polito.it/ lioy/01gsd/x509_6x.pdf

□ [16] http://www.di.unisa.it/ ads/corso-security/www/CORSO-0102/sito$_s$*icuro*/*SSL.htm*

□ *http* : //*www.crittologia.eu*/*critto*/*rsa*/*dh.html*

□ [17] http://www.ilsoftware.it/articoli.asp?tag=Disattivare-SSL-30-e-proteggere-i-dati-personali-dall-attacco-POODLE$_1$1420

http : *//www.rootsec.it/attacchi* − *mitm* − *su* − *connessioni* − *cifrate* − *con* − *sslstrip/*

Ringraziamenti