SECURE SOCKET LAYER

FEDERICO REALI

SOMMARIO. In questo articolo vengono esposte le principali caratteristiche del protocollo SSL. Esso è stato introdotto sin dal 1994 e rappresenta una delle soluzioni più utilizzata, più sicura e più affidabile, di trasporto *end-to-end* su TCP.

1. INTRODUZIONE

Il protocollo TCP fornisce un affidabile canale di comunicazione tra due nodi, esso permette di individuare quando vengono persi dei pacchetti, quando essi arrivano corrotti e non tiene conto dei dati ripetuti. Tuttavia implementando un autenticazione basata su indirizzo, può essere soggetto ad attacchi di *impersonation*. Inoltre tale protocollo manca di garanzie per quanto concerne autenticazione, integrità e confidenzialità.
SSL (Secure Socket Layer) è un protocollo aperto, non proprietario, sviluppato da Netscape Communication Corporation che nasce proprio con l'intento di proteggere il traffico del World Wide Web.
Nel tempo sono state presentate diverse versioni di tale protocollo:

1994: Versione 1 che presentava diversi problemi e mai pubblicata;
1994: Versione 2 implementata in Netscape Navigator 1;
1996: Versione 3 implementata in Netscape Navigator 3;
1999: TLS (Transport Layer Security) evoluzione di SSL proposta dal IETF, che formalizza il protocollo.[1] Nel tempo è stata sottoposta a diverse migliorie, fino alla RFC [2] 3268 che supporta tutte le più moderne tecnologia per la crittografia simmetrica (AES[3], IDEA[4] , DES e 3DES).

TLS garantisce la sicurezza del collegamento mediante tre funzionalità fondamentali:

Privatezza del collegamento: Per assicurare un collegamento sicuro tra i due soggetti coinvolti nella comunicazione, i dati vengono protetti attraverso algoritmi di crittografia a chiave simmetrica;
Autenticazione: Per assicurare l'identità dei soggetti coinvolti si possono utilizzare sia crittografia a chiave pubblica, sia certificati da parte del server e/o del client;
Affidabilità: Il livello di trasporto include un controllo di integrità sul messaggio basato su Message Autentication Code, utilizzando funzioni hash sicure come SHA e MD-5.

[1]Internet Engineering Task Force, comunità aperta di tecnici specialisti e ricercatori interessati alla evoluzione tecnica e tecnologica di internet.
[2]Request for Comments.
[3]Advanced Encryption Standard.
[4]International Data Encryption Algorithm.

Al momento è uno dei protocolli più utilizzati per la trasmissione sicura di dati e si trova in tutti i più noti browser.
Sebbene sia nato per rendere sicuro HTTP, è impiegato anche per altri protocolli come POP3, IMAP e FTP.

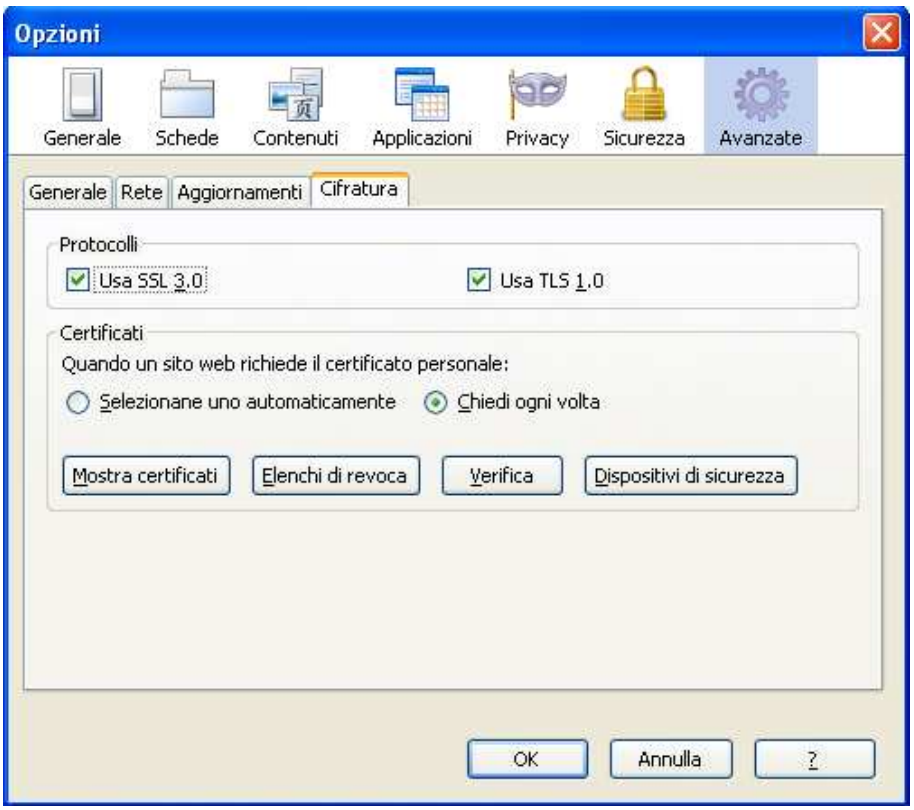

FIGURA 1. Opzioni per SSL su Firefox

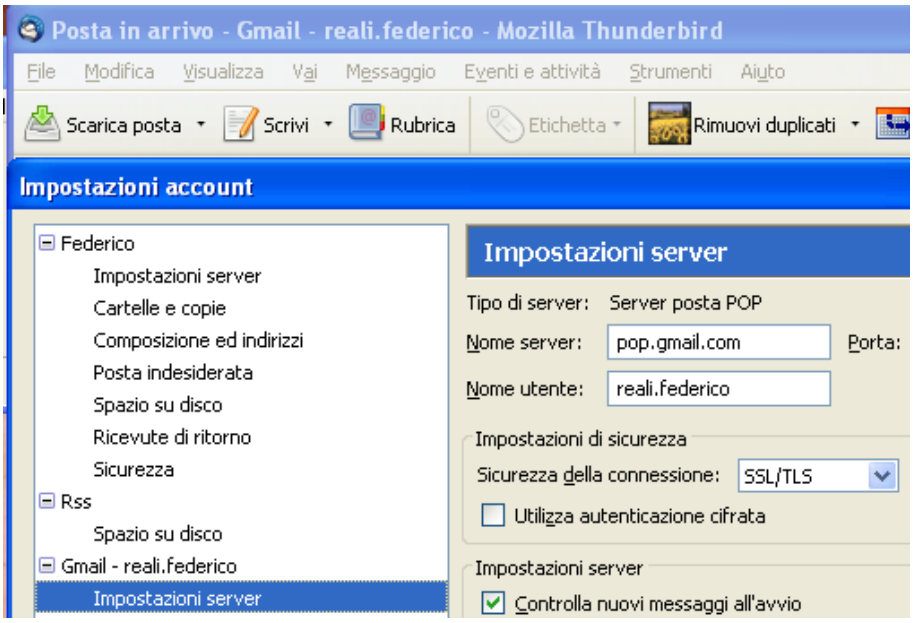

FIGURA 2. Opzioni per SSL su Thunderbird

Il protocollo SSL/TLS molto diffuso anche grazie al protocollo HTTPS (Hypertext Transfer Protocol over Secure Socket Layer) molto utilizzato nel web per lo scambio di dati sensibili, come il login ad un servizio, pagamenti di e-commerce o servizi di home banking.

FIGURA 3. Login su Amazon.co.uk con Google Chrome

FIGURA 4. Login su SOL unipg con Firefox

FIGURA 5. Login su Ebay con Opera

2. IL PROTOCOLLO

SSL/TLS è un protocollo composto da due layer (o strati), che si vanno ad inserire, nello stack dell'Internet Protocoll tra il layer del trasporto e il layer delle applicazioni. Tale posizionamento permette a SSL di ereditare le proprietà garantite da TCP e quindi di non trattare la gestione dei dati.
Il layer di SSL a livello più basso è formato dal protocollo SSL Record, che fornisce servizi di sicurezza di base, ai vari protocolli di livello più alto, come HTTP e FTP. Ci sono invece tre protocolli nel layer a livello più alto che sono definiti come parte di SSL: Handshake, Change Cipher Spec e Alert, i quali sono usati nelle gestione degli scambi SSL.
Due importanti concetti da distinguere sono quello di connessione SSL e sessione SSL:

Connessione: E' una connessione a livello trasporto che fornisce un certo tipo di servizi. Per SSL tali connessioni sono relazioni *peer-to-peer*[5]. Ogni connessione è associata ad una sessione.

Sessione: Una sessione SSL è un associazione tra un client e un server. Le sessioni sono create attraverso il protocollo Handshake. Le sessioni definiscono una serie di parametri di sicurezza, che possono essere condivisi tra più connessioni. Le sessioni servono per evitare costose negoziazioni di nuovi parametri di sicurezza per ogni connessione.

[5]Generalmente per peer-to-peer (o P2P), cioè rete paritaria, si intende una rete di computer o qualsiasi rete informatica che non possiede nodi gerarchizzati come client o server fissi, ma un numero di nodi equivalenti (in inglese peer) che fungono sia da client che da server verso altri nodi della rete.

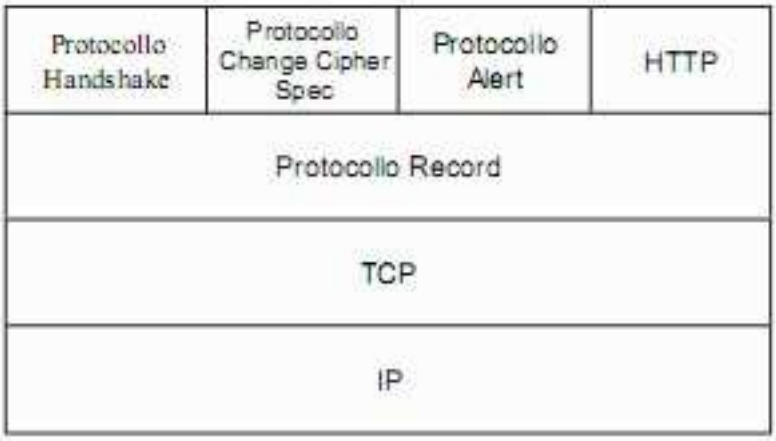

FIGURA 6. Stack del protocollo SSL

2.1. **SSL Record Protocol.** Questo è il protocollo che 'materialmente' prende i dati dai livelli superiori (compresi SSL Handshake, Change Cipher Spec e Alert), e li sottopone a trattamento di sicurezza, prima di passarli al livello sottostante (tipicamente TCP). Più precisamente, SSL Record fornisce due servizi per la connessione SSL:

Confidenzialità: Si usano a questo fine la chiave segreta e gli algoritmi di encryption simmetrica negoziati per la sessione tramite il protocollo Handshake;

Integrità del Messaggio: Il protocollo Handshake definisce anche una chiave segrta condivisa, che SSL Record usa per calcolare un opportuno Message Autentication Code (MAC).

I dati dai layer superiori del protocollo vengono prima frammentati in blocchi, poi compressi (se specificato), dopodiché viene calcolato il MAC sui dati compressi. A questo punto il messaggio compresso più il MAC vengono cifrati, e per ultimo viene apposto l'header[6] del protocollo SSL Record. Le FIGURE 7 e 8 riportano sia il funzionamento del SSL Record che la struttura interna del messaggio, dove il campo *Content Type* indica il tipo di dati contenuti nel frammento blocco, ovvero il protocollo che verrà usato per elaborare tali dati; il campo *Compressed Length* contiene la lunghezza del blocco di plaitext (compresso o meno) contenuto nel messaggio.

2.2. **Alert Protocol.** Il protocollo alert è usato per trasportare messaggi di allarme, compressi e cifrati. Ogni messaggio di questo protocollo consiste di due byte. Il primo byte rappresenta la gravità dell'allarme spedito (*warning* oppure *fatal*); il secondo byte contiene un codice che indica la motivazione del messaggio. Nel caso in cui il livello di gravità sia fatal, la connessione SSL viene terminata immediatamente.

2.3. **Change Cipher Spec Protocol.** Il protocollo Change Cipher Spec è il più semplice dei protocolli utilizzati da SSL. Esso consiste in un singolo messaggio, costituito da un solo byte con valore 1. L'unica funzione di questo messaggio è

[6]Parte del pacchetto che contiene informazioni di controllo necessarie al funzionamento della rete, posizionati solitamente all'inizio del pacchetto.

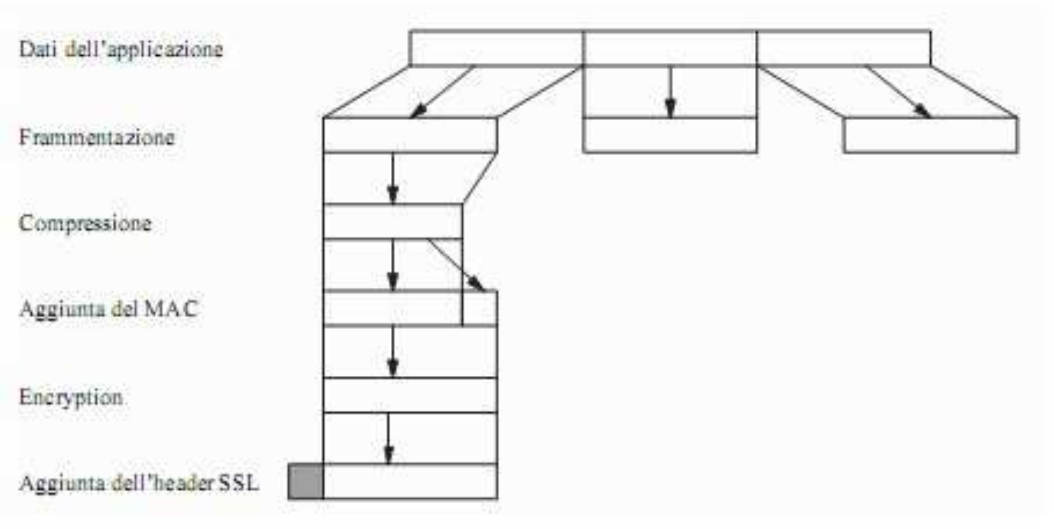

FIGURA 7. Operazioni del Record Protocol SSL

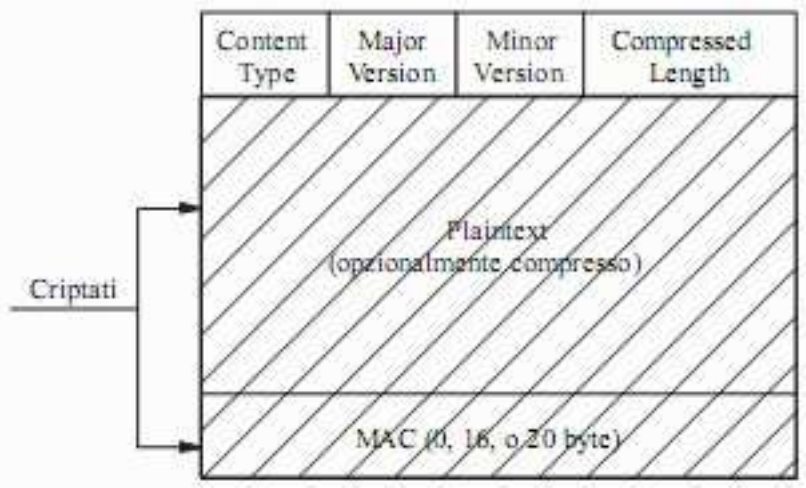

FIGURA 8. Formato di un messaggio SSL Record

fare in modo che siano aggiornate le informazioni riguardanti la suite di algoritmi di cifratura da utilizzare nella connessione, nel caso questi siano stati rinegoziati a livello di sessione tramite Handshake.

2.4. **Handshake Protocol.** E' di gran lunga il più complesso e permette al server e al client di:

- Autenticarsi a vicenda, negoziando il metodo di autenticazione;
- Negoziare un algoritmo di encryption;
- Negoziare un algoritmo per calcolare il MAC;
- Stabilire le chiavi segrete usate per cifrare i dati contenuti nei record SSL, negoziando il metodo per farlo.

Il protocollo Handshake viene usato prima della trasmissione dei dati delle applicazioni. La FIGURA 9 riporta lo scambio di messaggi tra client e server, le componenti tra parentesi sono opzionali.

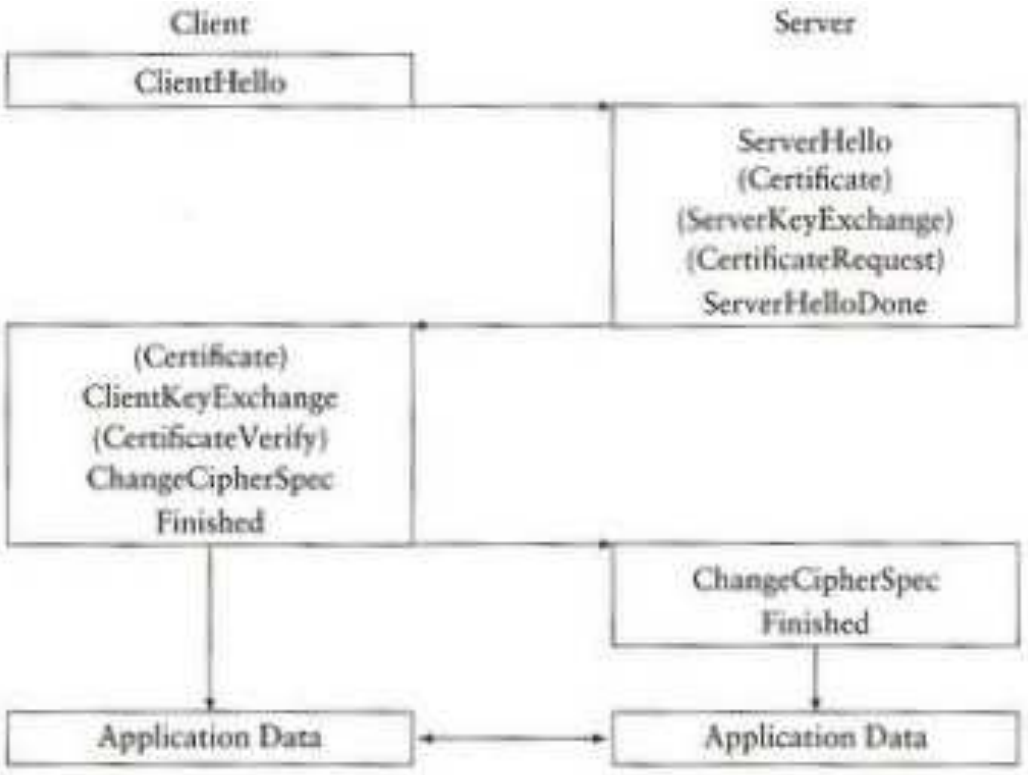

FIGURA 9. Il protocollo SSL Handshake

Prima di analizzare dettagliatamente il protocollo riportiamo in dettagli i parametri che possono essere riportati nei messaggi (FIGURA 10) e analiziamo i possibili parametri del messaggio *client-hello* o *server-hello* :

Tipo di messaggio	Parametri
hello_request	null
client_hello	versione, random, session id, suite di cifratura, metodo di compressione
server_hello	versione, random, session id, suite di cifratura, metodo di compressione
certificate	catena di certificati X.509v3
server_key_exchange	parametri, firma
certificate_request	tipo, authorities
server_done	null
certificate_verify	firma
client_key_exchange	parametri, firma
finished	valore hash

FIGURA 10. Tipi di messaggio e relativi parametri di SSL Handshake

- *Versione*: versione più recente di SSL conosciuta;

- *Random*: valore pseudocasuale generato localmente, usato come nonce durante la fase di scambio delle chiavi, per evitare attacchi di tipo *replay*;
- *Session ID*: identificatore di sessione a lunghezza variabile, il cui utilizzo verrà visto in seguito;
- *Suite di cifratura*: la lista contenente le combinazioni degli algoritmi crittografici supportati, in ordine decrescente di preferenza. Ogni elemento della lista specifica un metodo di cifratura (*CipherSpec*) e uno per lo scambio di chiavi. In particolare i metodi consentiti per lo scambio di chiavi sono:
  - RSA: Chiavi segrete cifrate con la chiave pubblica del ricevente, autenticate tramite certificato;
  - Fixed Diffie-Hellman: Algoritmo Diffie-Hellman in cui il certificato del server contiene i parametri pubblici Diffie-Hellman forniti da una CA (Certification Autority). Il client fornisce i propri parametri in un certificato oppure in un messaggio durante lo scambio di chiavi;
  - Ephemeral Diffie-Hellman: Viene utilizzato per creare chiavi segrete temporanee usando parametri pubblici *one-time*, scambiati e firmati usando RSA o simili. E' il metodo ritenuto più sicuro.
  - Anonymus Diffie-Hellman: Come Diffie-Hellman classico ma entrambe le parti spediscono i propri parametri pubblici senza autenticazione.
  - Fortezza: Un metodo poco usato nella pratica.
- *Metodo di compressione*: Lista di metodi di compressione supportati.

La figura riporta la serie di azioni che caratterizzano il protocollo di Handshake.
La comunicazione tra due nodi si divide in quattro fasi:
(Le figure riportano un possibile scambio di messaggi tra client e server)

Fase 1: Stabilire le specifiche per la sicurezza.

Questa fase è utilizzata per iniziare una connessione logica e stabilire le specifiche di sicurezza che le saranno associate. Lo scambio è iniziato dal client, che spedisce un messaggio *client-hello* con cui informa il server sugli algoritmi di cifratura e di compressione supportati, e spedisce un identificatore di sessione: un valore diverso da zero di tale parametro indica la volontà del client di aggiornare i parametri di una connessione esistente o di avviarne una nuova, mentre un valore uguale a zero invece indica la volontà di stabilire una nuova connessione in una nuova sessione. Dopodiché attende dal server un messaggio *server-hello*, del tutto simile a quello spedito dal client. L'unica particolarità sta nell'identificatore di sessione: se quello del client era diverso da zero lo stesso sarà per questo identificatore, altrimenti esso conterrà il nuovo valore generato dal server per questa sessione;

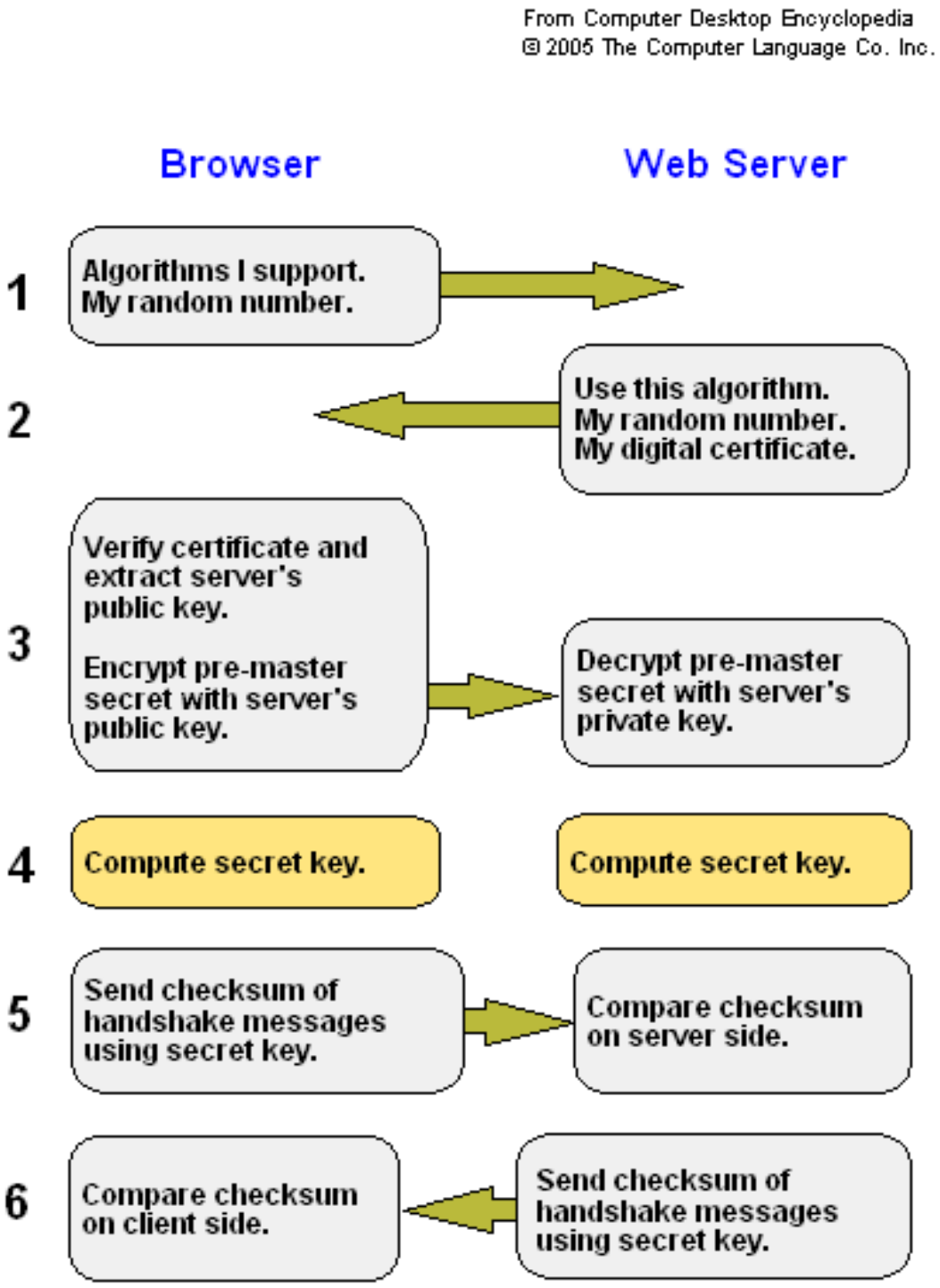

FIGURA 11. Azioni del Server e del Client in SSL Handshake

Fase 2: Autenticazione del server e scambio delle chiavi.
E' una fase in cui solo il server spedisce messaggi. Inizia con un messaggio *certificate*, necessario solo se si utilizza un metodo di negoziazione della chiave diverso da Anonymous Diffie-Hellman. A questo punto segue un messaggio *server-key-exchange*, che non è richiesto solo nel caso in cui sia stato

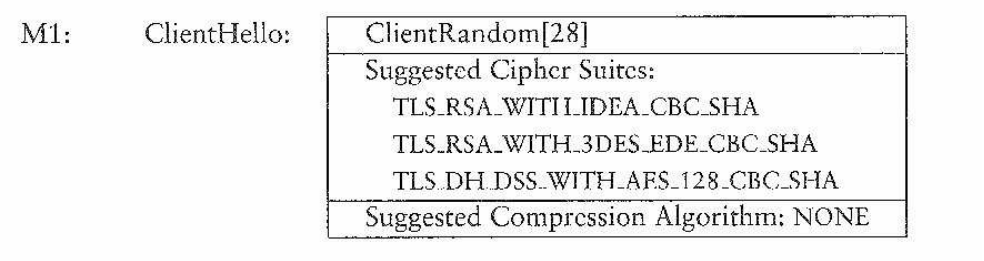

FIGURA 12. Messaggio della **Fase 1**

mandato un messaggio certificate con i parametri Fixed Diffie-Hellman oppure venga usato RSA per lo scambio di chiavi. La firma contenuta in questo messaggio è ottenuta calcolando un valore hash sui parametri di Diffie-Hellman o RSA e sui nonce presenti nei messaggi *hello* iniziali (per evitare attacchi di tipo replay), e cifrando tale valore con la chiave privata del server. Quello che segue è un messaggio di *certificate-request*, per richiedere l'autenticazione del client e può essere spedito solo da un server che non usi Anonymous Diffie-Hellman. Infine viene spedito un messaggio *server-hello-done*, che chiude la seconda fase.

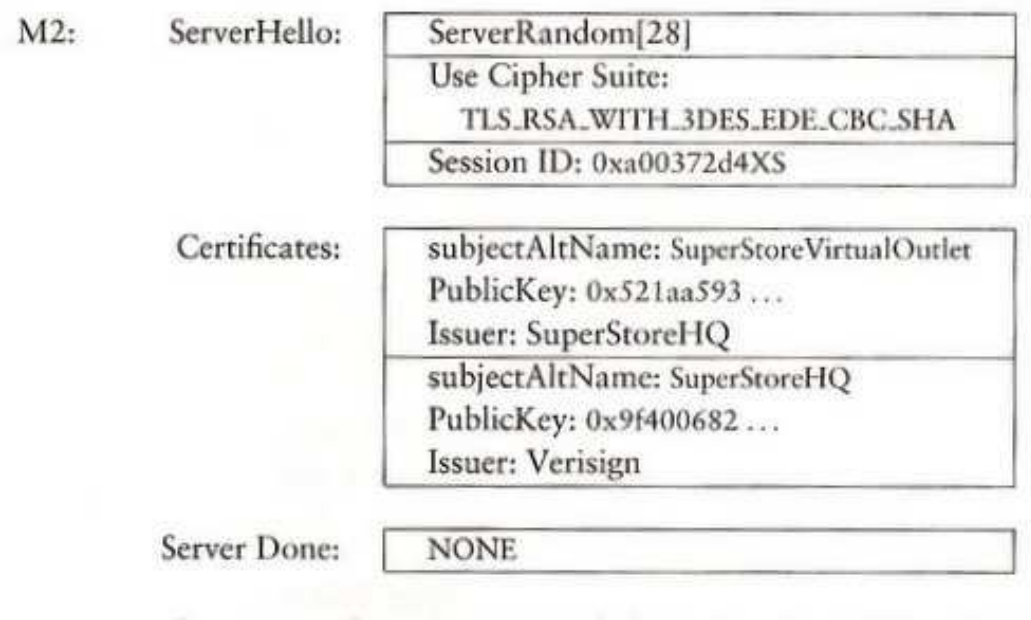

FIGURA 13. Messaggi della **Fase 2**

Fase 3: Autenticazione del client e scambio delle chiavi.
In questa fase il client procede alla verifica della validità del certificato del server e dei parametri del messaggio *server-hello*. Se tutto va bene il client spedisce un messaggio *certificate*, nel caso sia stato richiesto. Dopodiché spedisce un *client-key-exchange*, per lo scambio della chiave: esso contiene i parametri necessari, a seconda del tipo di algoritmo utilizzato. Vediamo alcuni esempi:

- Se l'algoritmo usato è RSA il client genera un *pre-master secret* e lo cifra con la chiave pubblica del certificato del server o con la chiave temporanea RSA appartenente al messaggio *server-key-exchange*. Il suo utilizzo verrà trattato in seguito;
- Se l'algoritmo è Ephemeral o Anonymous Diffie-Hellman, vengono spediti i parametri necessari;
- Se l'algoritmo è Fixed Diffie-Hellman, i parametri necessari vengono spediti in un messaggio *certificate*, quindi questo messaggio è vuoto.

Infine il client può trasmettere un messaggio *certificate-verify* per fornire la verifica esplicita del proprio certificato. Questo messaggio contiene una firma, costituita dalla cifratura di un valore hash calcolato sui precedenti messaggi, con la propria chiave privata, in modo che sia possibile verificare che la chiave privata appartiene al certificato del client. Nel calcolare il valore hash viene utilizzato un valore chiamato *master secret*, il cui calcolo è spiegato in seguito.

M3:	A: ClientKeyExchange:	RSA_Encrypt(ServerPublicKey,PreMasterSecret)
	B: ChangeCipherSpec:	NONE
	C: Finished	MD5(M1 || M2 || M3A) SHA(M1 || M2 || M3A)

FIGURA 14. Messaggi della **Fase 3**

Fase 4: Fine.

Questa fase completa la creazione di una connessione sicura. Il client spedisce un messaggio *change-Cipher-spec*, agganciando le specifiche riguardo ai metodi di encryption usati (*CipherSpec*), contenute anche nella *Suite di cifrature* del messaggio *client-hello* (*server-hello*), secondo quanto concordato in precedenza. Dopodiché manda un messaggio *finished* per terminare la comunicazione. Tale messaggio contiene la concatenazione di due valori hash, calcolati sul master secret e sui messaggi precedenti, uno con SHA-1 e l'altro con MD-5. Il server come risposta, fa le stesse cose e termina lo scambio di messaggi. A questo punto la connessione è stabilita.

M4:	A: ChangeCipherSpec:	NONE
	B: Finished	MD5(M1 || M2 || M3A || M3C) SHA(M1 || M2 || M3A || M3C)

FIGURA 15. Messaggi della **Fase 4**

Creazione del Master Secret. Il master secret condiviso è un valore temporaneo a 48 bit calcolato durante lo scambio delle chiavi. La sua creazione avviene in due passi. Prima viene scambiato un *pre-master secret* come visto in precedenza nella **Fase 3** del protocollo di Handshake. Dopodiché entrambe le parti ottengono il

master secret concatenando tre valori hash calcolati sul pre-master secret e sui due nonce scambiati inizialmente.
A questo punto sia il server che il client sono in possesso della chiave da utilizzare per la crittografia simmetrica, il *Master Secret*, e quindi possono avviare uno scambio di dati sicuro criptando i dati attraverso l'algoritmo specificato.

Riferimenti bibliografici

[1] Giorgio Faina: *Appunti alle lezioni di Laboratorio di Teoria dell' Informazione e Codici.*
[2] Dieter Gollmann: *Computer Security*, John Wiley and Sons, Ltd, Second Edition, 2006.
[3] Williwam Stalling: *Cryptography and Network Security, Principles and Practice*, Pearson Education LTD, Fourth Edition, 2006
[4] *Wikipedia the free Enciclopedia* : `http://it.wikipedia.org/wiki/Wiki`