HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA VIỄN THÔNG 1

TIỂU LUẬN MÔN
AN NINH MẠNG THÔNG TIN

Đề tài:
TÌM HIỂU VỀ GIAO THỨC BẢO MẬT SSL/TLS

Giảng viên : Nguyễn Thanh Trà

LỜI NÓI ĐẦU

Ngày nay, các thuật ngữ như thương mại điện tử (E-Commerce), kinh doanh điện tử (E-Business) và chính phủ điện tử (E-Government) có mặt khắp nơi. Khi các ngành công nghiệp số bùng nổ, việc đề cập đến các yêu cầu bảo mật nghiêm ngặt phải được đáp ứng theo từng nhu cầu hay từng cách tiếp cận về bảo mật. Dữ liệu trong khi truyền giữa người gửi và người nhận là lúc dễ bị tấn công nhất. Hầu hết các phương pháp tấn công đều nhằm vào dữ liệu, đều thực hiện trong quá trình giao dịch điện tử.

Do đó, bảo mật kênh truyền dữ liệu trong việc thực hiện các giao dịch thương mại điện tử là rất quan trọng. Hiện nay, bảo mật kênh truyền chủ yếu tập trung vào các giao thức mã hóa. Một số giao thức bảo mật kênh truyền dữ liệu như: Giao thức SSL (Secure Socket Layer), giao thức SET (Secure Electronic Transaction), giao thức WEP (Wireless Encryption Protocol)…

SSL được thiết kế cho những giải pháp truy nhập từ xa và không cung cấp những kết nối site-to-site. SSL VPNs cung cấp vấn đề bảo mật truy cập đầu tiên những ứng dụng web. Bởi vì SSL sử dụng trình duyệt web, điển hình là những người sử dụng không phải chạy bất kỳ phần mềm client đặc biệt nào trên những máy tính của họ.

Để bảo vệ những thông tin mật trên mạng Internet hay bất kỳ mạng TCP/IP nào, SSL (Secure Sockets Layer) đã kết hợp những yếu tố sau để thiết lập được một giao dịch an toàn. Đó là khả năng bảo mật thông tin, xác thực và toàn vẹn dữ liệu đến người dùng. SSL được tích hợp sẵn vào các Browser và WebServer, cho phép người sử dụng làm việc với các trang Web ở chế độ an toàn.

Trong giới hạn đề tài, nhóm 8 tìm hiểu về giao thức bảo mật SSL. Nội dung của đề tài bài tiểu luận trình bày gồm tổng quan về giao thức SSL, cấu trúc và cách thức hoạt động. Qua đó đánh giá việc sử dụng giao thức SLL. Nội dung được chia thành 3 chương.

CHƯƠNG 1. TỔNG QUAN VỀ GIAO THỨC SSL.

CHƯƠNG 2. CẤU TRÚC VÀ HOẠT ĐỘNG CỦA SSL.

CHƯƠNG 3. ĐÁNH GIÁ VIỆC SỬ DỤNG GIAO THỨC SSL.

Do thời gian hạn chế, nội dung bài tiểu luận không tránh khỏi những thiếu sót, nhóm 8 rất mong nhận được sự chỉ đạo và góp ý của thầy cô.

BẢNG PHÂN CHIA CÔNG VIỆC

	TÊN SINH VIÊN	NỘI DUNG	CHI TIẾT CÔNG VIỆC
1		CHƯƠNG 1. TỔNG QUAN VỀ GIAO THỨC SSL 1.1. Giới thiệu chung về giao thức SSL. 1.2. Giới thiệu về giao thức SSL. 1.3. Kết luận chương 1:	- Word - Viết kết luận chương - Tổng hợp
2		CHƯƠNG 2. CẤU TRÚC VÀ CÁCH LÀM VIỆC CỦA SSL. 2.1 Cấu trúc của SSl 2.1.1 Giao thức bắt tay (Handshake Protocol) 2.1.2 Giao thức SSL Change Cipher Spec Protocol: 2.1.3 Giao thức SSL Alert: 2.1.4 SSL Record Layer:	- Word - Viết kết luận chương - Tổng hợp
3		CHƯƠNG 2. CẤU TRÚC VÀ CÁCH LÀM VIỆC CỦA SSL. 2.2 Hoạt Động Của Giao Thức SSL 2.3 Cơ chế hoạt động SSL 2.4 Tấn công và cách phòng chống 2.5 Kết luận chương 2 CHƯƠNG 3. ĐÁNH GIÁ GIAO THỨC SSL	- Word

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG

CHƯƠNG 1. TỔNG QUAN VỀ GIAO THỨC SSL

1.1. Giới thiệu chung về giao thức SSL.

1.1.1 Lịch sử phát triển của của giao thức SSL:

Một trong những điều khiến cuộc cách mạng internet trở nên khả thi là giao thức bảo mật cho phép mọi người làm những việc mới lạ như mua sách trực tuyến bằng cách thanh toán an toàn. Mọi chuyện bắt đầu khi Taher Elgamal , một nhà mật mã học người Ai Cập đồng thời là nhà khoa học chính của Netscape Communications, thúc đẩy sự phát triển của giao thức internet Lớp cổng bảo mật (SSL).

Netscape Communication đã giới thiệu SSL và một giao thức tương ứng với phiên bản đầu tiên của Netscape Navigator. Netscape Communication đã không tính phí các khách hàng của mình khi thực thi giao thức bảo mật. Sau đó, SSL trở thành giao thức nổi bật để cung cấp các dịch vụ bảo mật cho lưu lượng dữ liệu HTTP đã mất dần vị thế.

SSL được coi là giao thức bảo mật quan trọng lớp vận chuyển (Layer Transport) có tầm quan trọng cao nhất đối với sự bảo mật của các trình ứng dụng trên web. Nói chung, có một số khả năng bảo vệ bằng mật mã lưu lượng dữ liệu HTTP.

1.1.2. Các phiên bản của giao thức SSL

Cho đến bây giờ, có ba phiên bản của SSL:

- **Phiên bản SSL 1.0**:

Được sử dụng nội bộ chỉ bởi Netscape Communications. Phiên bản này đã chứa một số khiếm khuyết nghiêm trọng nên không được tung ra bên ngoài.

- **Phiên bản SSL 2.0:**

Vào năm 1996, Được kết nhập vào Netscape Communications 1.0 đến 2.x. Phiên bản 2.0 có một số điểm yếu liên quan đến sự hiện thân cụ thể của cuộc tấn công của đối tượng trung gian. Trong một nỗ lực nhằm dùng sự không chắc chắn của công chúng về bảo mật của SSL. Khi đó Microsoft cũng đã giới thiệu giao thức PCT (Private Communication Technology) cạnh tranh trong lần tung ra Internet Explorer.

- **Phiên bản SSL 3.0:**

Tháng 3/1996, thông số kỹ thuật SSL 3.0 đã được tung ra chính thức. Netscape Communications đã đáp lại thách thức PTC của Microsoft bằng cách giới thiệu SSL 3.0 nhằm giải quyết các vấn đề trong SSL 2.0. Nó được thực thi trong tất cả các trình duyệt

chính, ví dụ Microsoft Internet Explorer 3.0, Netscape Navigator 3.0 (và các phiên bản cao hơn).

1.2. Giới thiệu về giao thức SSL.

Giao thức SSL được hình thành và phát triển đầu tiên năm 1994 bởi nhóm nghiên cứu Netscape dẫn dắt bởi Elgammal và ngày nay đã trở thành chuẩn bảo mật thực hành trên mạng Internet.

1.2.1. Định nghĩa giao thức SSL

SSL (Secure Socket Layer) là giao thức đa mục đích được thiết kế để tạo ra các giao tiếp giữa hai chương trình ứng dụng trên một cổng định trước (socket 443) nhằm mã hoá toàn bộ thông tin đi/đến, mà ngày nay được sử dụng rộng rãi cho giao dịch điện tử như truyền số hiệu thẻ tín dụng,mật khẩu, số bí mật cá nhân (PIN) trên Internet.

SSL hiện nay là giao thức bảo mật rất phổ biến trên Internet trong các hoạt động thương mại điện tử (E-Commerce). SSL được thiết kế như là một giao thức riêng cho vấn đề bảo mật có thể hỗ trợ cho rất nhiều ứng dụng. Giao thức SSL tổ hợp nhiều giải thuật mã hóa nhằm đảm bảo quá trình trao đổi thông tin trên mạng được bảo mật.Việc mã hóa dữ liệu diễn ra một cách trong suốt, hỗ trợ nhiều giao thức khác chạy trên nền giao thức TCP

1.2.2. Những yếu tố thiết lập giao dịch an toàn:

Để đảm bảo tính bảo mật thông tin trên Internet hay bất kì mạng TCP/IP nào thì SSL ra đời kết hợp với những yếu tố sau để thiết lập giao dịch an toàn:

- **Xác thực:** Đảm bảo tính xác thực của trang mà bạn sẽ làm việc ở đầu kia của kết nối. Như vậy, các trang Web cũng cần phải kiểm tra tính xác thực của người sử dụng.
- **Mã hoá:** đảm bảo thông tin không thể bị truy cập bởi đối tượng thứ ba.Để loại trừ việc nghe trộm những thông tin " nhạy cảm" khi nó được truyền qua Internet.
- **Toàn vẹn dữ liệu:** đảm bảo thông tin không bị sai lệch và nó phải thể hiện chính xác thông tin gốc gửi đến.

1.3 Kết luận chương

Chương 1 khái quát lịch sử hình thành và có cái nhìn tổng quan về giao thức. Mỗi một phiên bản của SSL là một lần nâng cấp đáp ứng được những yêu cầu, vấn đề cũ mà phiên bản trước gặp phải. Tuy vậy vẫn cần có yếu tố để thiết lập giao dịch an toàn thông qua SSL đó những yếu tố như xác thực, mã hóa, tính toàn vẹn dữ liệu.

CHƯƠNG 2. CẤU TRÚC VÀ CÁCH LÀM VIỆC CỦA SSL.

2.1 Cấu trúc của SSl

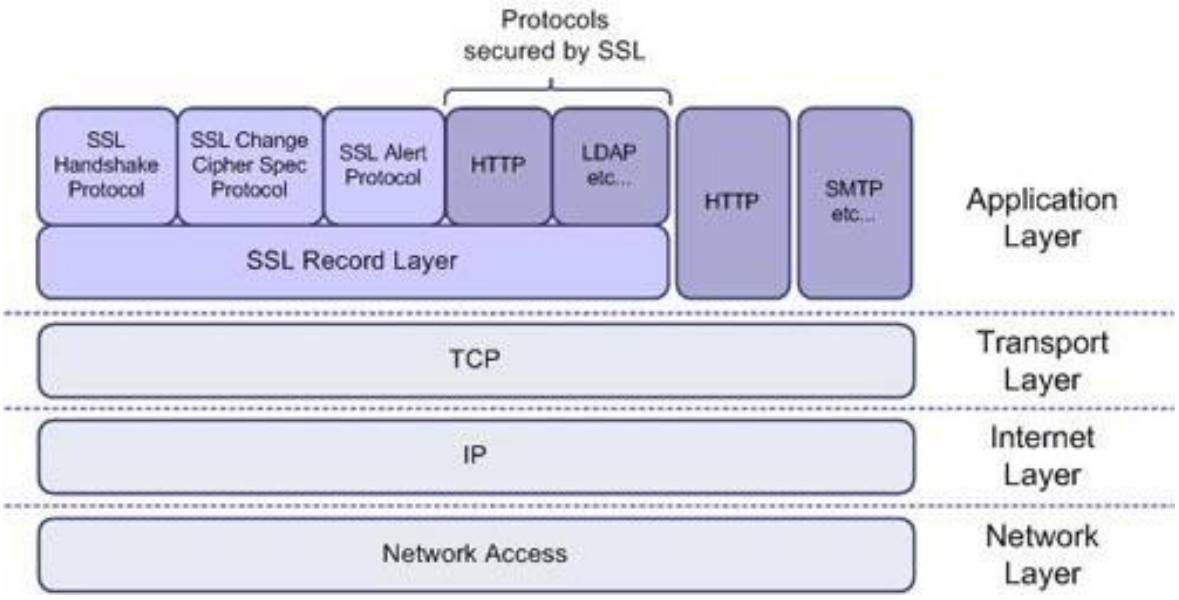
SSL là giao thức tầng (Layer Protocol), giao thức SSL gồm hai phần chính:

- SSL Record Protocol
- Một số giao thức con được sắp xếp trên nó.

Các giao thức con SSL được sắp xếp lớp trên SSL Record Protocol để cung cấp sự hỗ trợ cho việc quản lý session SSL và thiết lập nối kết. Như vậy, cấu trúc SSL gồm 4 giao thức con sau:

- Giao thức SSL Handshake
- Giao thức SSL Change Cipher Spec
- Giao thức SSL Alert
- Giao thức SSL Record Layer

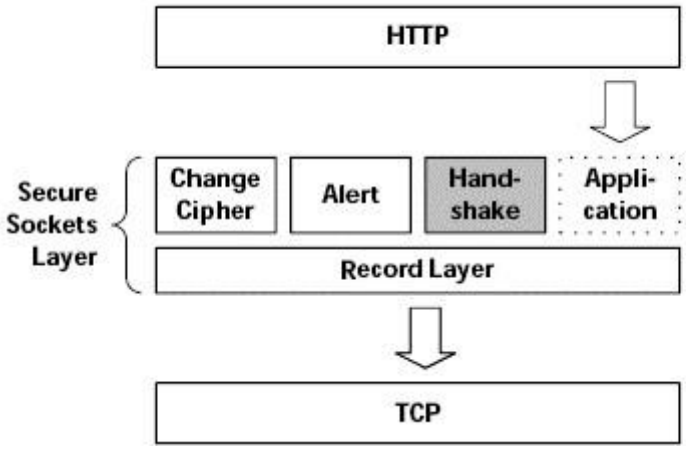
Vị trí của các giao thức trên, tương ứng với mô hình TCP/IP được minh hoạ theo biểu đồ sau:



Hình 2.1. Cấu trúc SSL tương ứng trên TCP/IP.

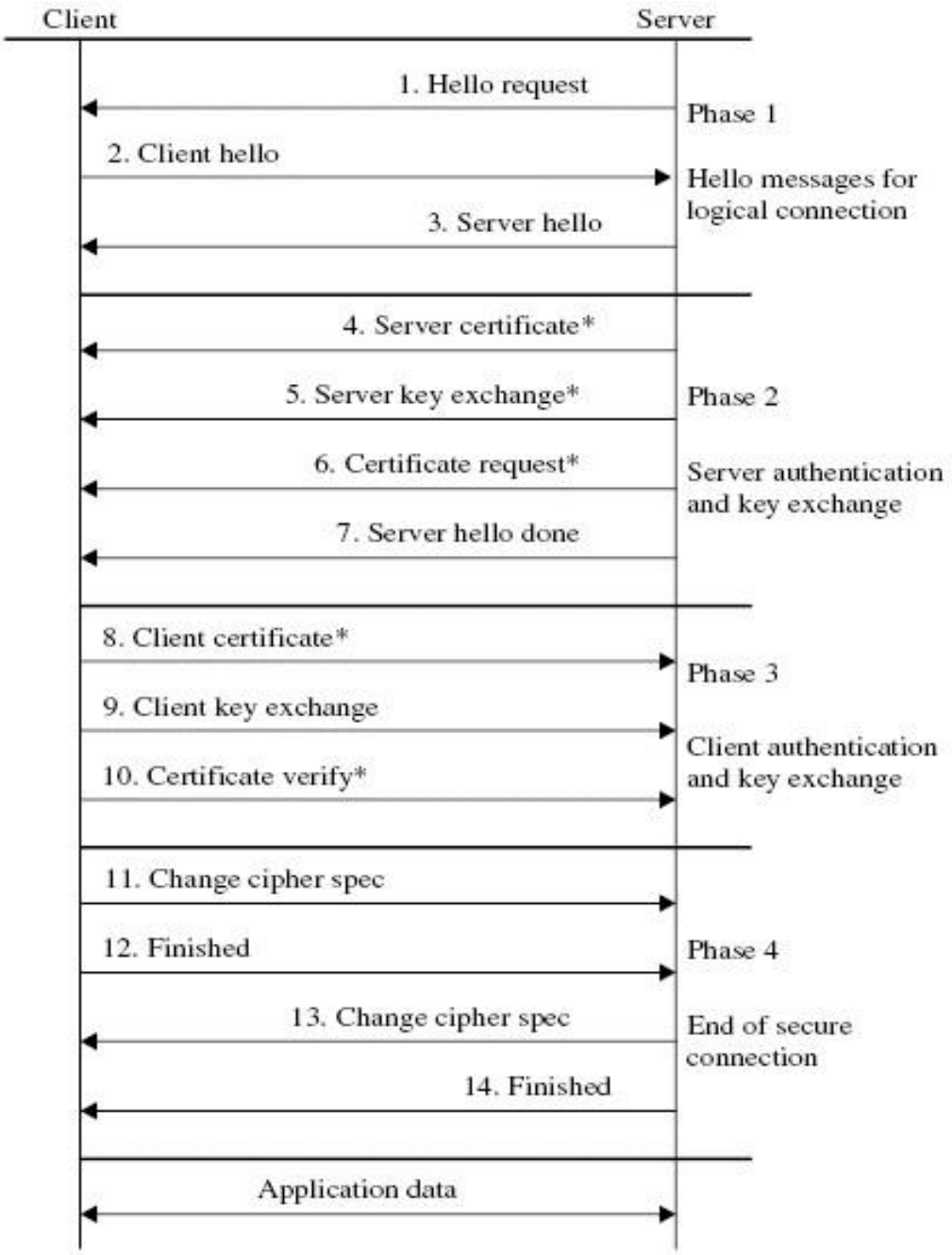
2.1.1 Giao thức bắt tay (Handshake Protocol)

SSL Handshake Protocol là một phần quan trọng của SSL, nó cung cấp ba dịch vụ cho các kết nối SSL giữa client và server. Handshake Protocol cho phép client/server thống nhất về phiên bản giao thức, xác thực mỗi bên bằng cách thi hành một MAC và thoả thuận về một thuật toán mã hoá và các khoá lập mã cho việc bảo vệ các dữ liệu gửi đi trong một SSL record trước khi giao thức ứng dụng truyền đi hay nhận được byte dữ liệu đầu tiên.



Hình 2.2. Vị trí giao thức bắt tay

Giao thức bắt tay (Handshake Protocol) bao gồm một dãy các thông điệp được trao đổi bởi client và server. Hình 2.2 minh hoạ sự trao đổi các thông điệp handshake cần thiết để thiết lập một kết nối logic giữa client và server. Nội dung và ý nghĩa mỗi thông điệp được mô tả chi tiết trong các phần sau.



Hình 2.3. Tiến trình bắt tay

" * " Cho biết đây là tuỳ chọn hoặc là các thông điệp phụ thuộc tuỳ vào tình huống cụ thể sẽ không gửi đi.

Tiến trình hoạt động của giao thức bắt tay có thể được mô tả tóm tắt như sau:

1, Client gửi tới server số phiên bản SSL của client, các tham số của thuật toán mã hoá, sinh dữ liệu ngẫu nhiên (đó chính là digital signature) và các thông tin khác mà server cần để thiết lập kết nối với client.

2, Server gửi tới client số phiên bản SSL của server đang dùng, các tham số của thuật toán mã hoá, sinh dữ liệu ngẫu nhiên và các thông tin khác mà client cần để thiết lập kết nối với server có sử dụng SSL. Server cũng gửi chứng chỉ (certificate) của mình tới client, nếu client yêu cầu tài nguyên của server mà cần sự xác thực client thì server sẽ yêu cầu chứng chỉ của client.

3, Client sử dụng một số thông tin mà server gửi đến để xác thực server. Nếu như server không được xác thực thì người sử dụng sẽ được cảnh báo và kết nối không được thiết lập. Còn nếu như xác thực được server thì phía client sẽ thực hiện tiếp bước 4.

4, Sử dụng tất cả các thông tin được tạo ra trong giai đoạn bắt tay ở trên, client (cùng với sự cộng tác của server và phụ thuộc vào thuật toán được sử dụng) sẽ tạo ra **premaster secret** cho phiên làm việc, mã hoá bằng khoá công khai (public key) mà server gửi đến trong chứng chỉ ở bước 2, và gửi đến server.

5, Nếu server có yêu cầu xác thực client (tuỳ chọn trong quá trình bắt tay), client sẽ ký trên dữ liệu, dữ liệu này là duy nhất đối với quá trình bắt tay và đều được lưu trữ bởi client và server. Trong trường hợp này, client sẽ gửi cả dữ liệu được ký, chứng chỉ số của mình cùng với **premaster secret** đã được mã hoá tới server.

6, Nếu server yêu cầu xác thực client, server sẽ cố gắng để xác thực client. Trường hợp client không được xác thực, phiên làm việc sẽ bị ngắt. Còn nếu client được xác thực thành công, server sẽ sử dụng khoá riêng (private key) để giải mã **premaster secret**, sau đó thực hiện một số bước để tạo ra **master secret**.

7, Client và server sử dụng **master secret** để tạo ra các **session key (khoá phiên)**, đó chính là các khoá đối xứng được sử dụng để mã hoá và giải mã các thông tin trong phiên làm việc và kiểm tra tính toàn vẹn dữ liệu-xác định sự thay đổi về dữ liệu giữa thời điểm gửi và nhận.

8, Client sẽ gửi một thông điệp tới server thông báo rằng các thông điệp tiếp theo sẽ được mã hoá bằng **session key**. Sau đó gửi kèm theo một thông điệp riêng biệt xác định quá trình bắt tay phía server đã kết thúc.

9, Lúc này giai đoạn "bắt tay" đã hoàn thành, và phiên làm việc SSL bắt đầu. Cả hai phía client và server sẽ sử dụng các **session key** để mã hoá và giải mã thông tin trao đổi giữa hai bên, và kiểm tra tính toàn vẹn dữ liệu.

Khi client và server quyết định sử dụng lại phiên trước hoặc tạo một bản sao của phiên đang tồn tại (thay vì phải thoả thuận các tham số an toàn mới), thì luồng

thông điệp hoạt động như sau: client gửi một ClientHello có sử dụng Session ID của phiên được dùng lại server kiểm tra nơi lưu trữ phiên (session cache) tương ứng. Nếu có, server sẽ thiết lập lại kết nối dưới trạng thái phiên được chỉ định, server sẽ gửi một thông điệp ServerHello có giá trị Session ID giống hệt. Mỗi khi quá trình thiết lập lại được hoàn thành thì client và server có thể trao đổi dữ liệu lớp ứng dụng.

Ngược lại nếu Session ID không tìm thấy thì server sẽ tạo ra một Session ID mới, SSL client và server thực hiện quá trình bắt tay thông thường như ở trên.

a, Giai đoạn 1: Các thông điệp Hello cho kết nối logic

Client gửi một thông điệp ClientHello tới server, server phải đáp ứng lại bằng một thông điệp ServerHello, hoặc ngược lại nếu một lỗi xảy ra và kết nối sẽ thất bại. ClientHello và ServerHello được sử dụng để tăng tính an toàn giữa client và server. ClientHello và ServerHello thiết lập các thuộc tính: Protocol Version, Session ID, Cipher Suite và Compression Method. Ngoài ra hai giá trị ngẫu nhiên được tạo ra và được trao đổi là: ClientHello.random và ServerHello.random. Các thông điệp giai đoạn chào hỏi được sử dụng để trao đổi các thuộc tính thừa kế bảo mật giữa client và server.

- **Thông điệp HelloRequest**

Thông điệp này được gửi bởi server tại bất kỳ thời điểm nào, nhưng có thể bị client bỏ qua nếu Handshake Protocol đã đang thực hiện. Nếu client nhận một HelloRequest trong một trạng thái thoả thuận bắt tay thì client đơn giản là bỏ qua thông điệp này.

- **Thông điệp ClientHello**

Thông điệp ClientHello bắt đầu phiên truyền thông SSL giữa hai bên. Client sử dụng thông điệp này hỏi server để bắt đầu thoả thuận các dịch vụ bảo mật được sử dụng SSL. Sau đây là các thành phần quan trọng của một thông điệp ClientHello:

- Version: phiên bản SSL cao nhất mà client hỗ trợ. Phiên bản SSL hiện tại là 3.0. Chú ý rằng một server có thể thừa nhận rằng client có thể hỗ trợ tất các phiên bản SSL cao hơn và bao gồm giá trị của trường này. Ví dụ, nếu một client gửi một ClientHello với Version có giá trị 3 tới server chỉ hỗ trợ SSL version 2.0, server có thể đáp ứng bằng các thông điệp version 2.0 và nó hy vọng client hiểu. Client có thể quyết định tiếp tục với phiên SSL sử dụng version 2.0, hoặc có thể bỏ qua.
- Random: Một cấu trúc ngẫu nhiên được sinh ra, bao gồm một nhãn thời gian 32-bit và 28 byte được sinh bởi bộ sinh số ngẫu nhiên bảo mật. Nhãn thời gian-32 byte bao gồm thời gian và ngày, để chắc chắn rằng client không bao giờ sử dụng cùng một

giá trị ngẫu nhiên hai lần. Sử dụng phương pháp này để chống lại một sự sao chép bất hợp pháp các thông điệp SSL từ một client bất hợp pháp và sử dụng lại chúng để thiết lập một phiên giả mạo. 28 byte còn lại là một số ngẫu nhiên "bảo mật mật mã", sử dụng một công nghệ được biết đến như là "sinh số giả ngẫu nhiên" để tạo ra các số ngẫu nhiên.

- Session ID: một định danh phiên chiều dài biến. Một giá trị khác không thể hiện rằng client muốn cập nhật các tham số của kết nối đang tồn tại hoặc tạo một kết nối mới trong phiên này. Một giá trị bằng không thể hiện rằng client muốn thiết lập một kết nối mới trên một phiên làm việc mới.
- CipherSuite: đây là một danh sách bao gồm các sự kết hợp của các thuật toán lập mã được hỗ trợ bởi client, xếp tăng theo thứ tự tham chiếu. Mỗi phần tử của danh sách định nghĩa cả hai thuật toán trao đổi khoá và một CipherSpec.
- Compression Method: danh sách các phương pháp nén client hỗ trợ.

- **ServerHello**:

Khi server nhận được thông điệp ClientHello, nó gửi đáp ứng lại với một thông điệp ServerHello. Nội dung của ServerHello tương tự như ClientHello, tuy nhiên, cũng có một số điểm khác biệt quan trọng.

- Version: lưu giá trị phiên bản thấp nhất được chấp nhận bởi client trong ClientHello và phiên bản cao nhất được hỗ trợ bởi server. Version trong ServerHello quyết định phiên bản SSL mà kết nối sẽ sử dụng.
- Random: giá trị ngẫu nhiên được sinh ra bởi server, bốn byte đầu là nhãn thời gian (để tránh các giá trị ngẫu nhiên lặp lại); các byte còn lại sẽ được tạo bởi bộ sinh số ngẫu nhiên bảo mật lập mã.
- Session ID: trường Session ID của ServerHello có thể lưu một giá trị, không giống như trường Session ID trong ClientHello đã được nói tới. Giá trị này định danh duy nhất truyền thông SSL cụ thể hoặc session. Lý do chính cho việc định danh một phiên SSL cụ thể là để sau này có thể dùng lại. Nếu server không chỉ định ra phiên đã từng sử dụng, nó có thể bỏ qua trường Session ID trong thông điệp ServerHello.
- CipherSuite: quyết định các tham số lập mã, các thuật toán và các cỡ khoá, được sử dụng trong một phiên. Server phải chọn một bộ mã đơn từ danh sách mà client đưa cho trong thông điệp ClientHello.

- CompressionMethod: server sử dụng trường này để định danh việc nén dữ liệu sử dụng cho phiên. Một lần nữa, server phải lấy chúng từ danh sách đã được liệt kê trong ClientHello. Tuy nhiên, các phiên bản SSL hiện tại không định nghĩa bất kỳ phương pháp nén nào, vì thế trường này không có ý nghĩa thiết thực cụ thể.

b Giai đoạn 2: Xác thực Server và trao đổi khoá

Sau các thông điệp hello, server bắt đầu giai đoạn này bằng cách gửi chứng chỉ của nó nếu nó cần được xác thực. Thêm vào đó, một thông điệp ServerKey Exchange có thể được gửi nếu nó được yêu cầu. Nếu server đã được xác thực, nó có thể yêu cầu một chứng chỉ từ client, nếu nó thích hợp với bộ mã được chọn. Sau đó, server sẽ gửi thông điệp ServerHelloDone, thể hiện rằng giai đoạn thông điệp hello của handshake đã hoàn thành. Server sau đó sẽ đợi đáp ứng của client. Nếu server đã gửi một thông điệp yêu cầu chứng chỉ (Certificate request message), client phải gửi lại thông điệp chứng chỉ (certificate message)

- **Thông điệp ServerCertificate**

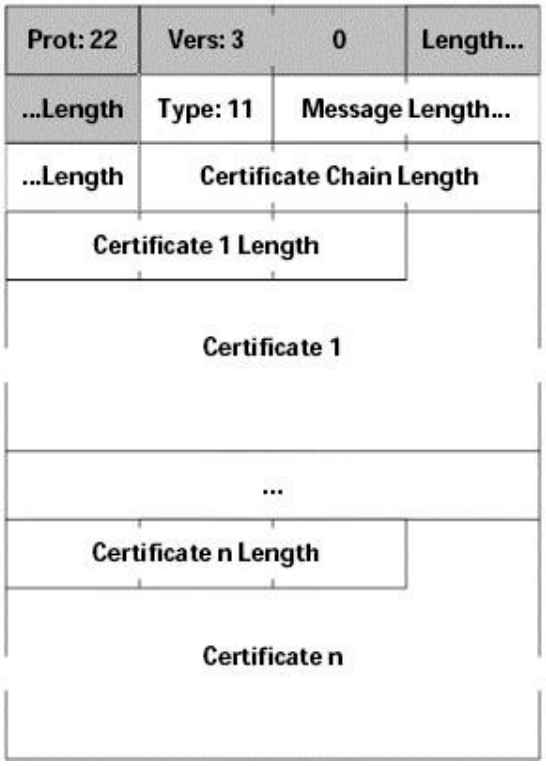
Nếu server đã được xác thực, nó phải gửi một chứng chỉ ngay sau thông điệp ServerHello. Kiểu chứng chỉ phải phù hợp với thuật toán chuyển đổi khoá của bộ mã (cipher suite) đã lựa chọn, và nó thường là một chứng chỉ X.509 v3. Nó phải chứa một khoá phù hợp với phương pháp chuyển đổi khoá (key exchange method). Thuật toán ký cho chứng chỉ phải giống như thuật toán cho khoá chứng chỉ (certificate key).

Thông điệp Certificate tương đối dễ hiểu, được minh hoạ trong hình 2.4

Kiểu thông điệp giao thức Handshake của nó là 11, và nó bắt đầu với kiểu thông điệp và chiều dài thông điệp bắt tay chuẩn. Phần thân của thông điệp bao gồm một dãy các chứng chỉ khoá công khai. Dãy này bắt đầu với 3 byte thể hiện chiều dài. (Giá trị chiều dài dãy luôn là 3 nhỏ hơn giá trị chiều dài thông điệp). Mỗi chứng chỉ trong dãy cũng bắt đầu với một trường 3 byte lưu cỡ của chứng chỉ. Thông điệp đầu tiên thể hiện toàn bộ chiều dài của dãy chứng chỉ. Phần sau đó thể hiện chiều dài của mỗi chứng chỉ với 3 byte ngay trước chứng chỉ đó.

Dãy các thông điệp cho phép SSL hỗ trợ các hệ chứng chỉ. Chứng chỉ đầu tiên trong dãy luôn là của bên gửi. Chứng chỉ tiếp theo là của bên cấp chứng chỉ cho chứng chỉ của bên gửi. Chứng chỉ thứ ba thuộc về CA cho quyền đó, và cứ tiếp tục như thế.

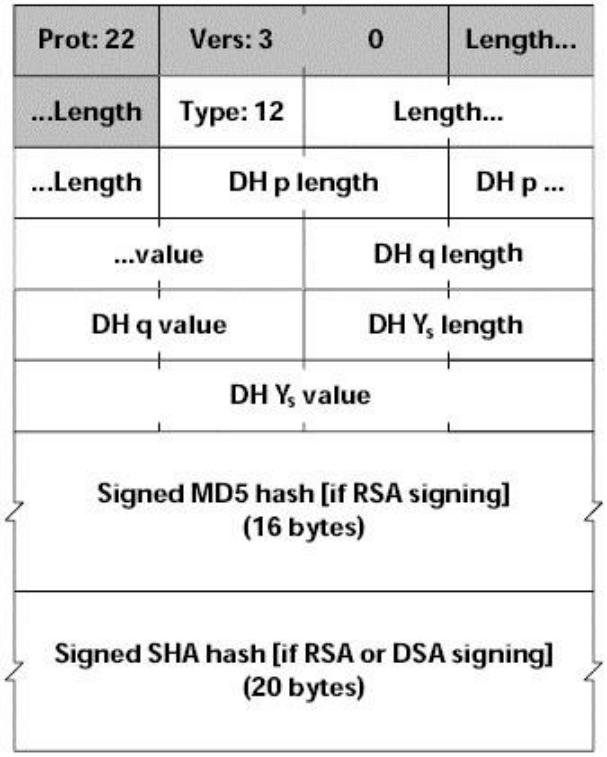
Dãy chứng chỉ sẽ tiếp tục cho tới khi nó tìm thấy một chứng chỉ cho quyền chứng chỉ gốc (root).



Hình 2.4. Thông điệp Certificate

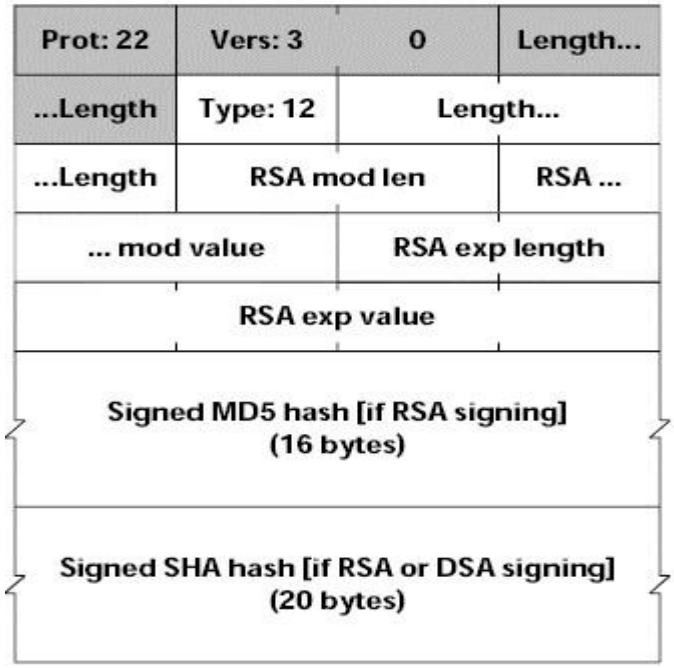
- **Thông điệp ServerKeyExchange**

Thông điệp ServerKeyExchange truyền thông tin khoá từ server tới client. Định dạng chính xác của nó tuỳ thuộc vào các thuật toán lập mã được sử dụng để trao đổi thông tin khoá. Các định dạng khác nhau phù hợp với các giao thức khoá Diffie-Hellman, RSA, và Fortezza được minh hoạ trong hình 2.5, 2.6, 2.7. Trong mọi trường hợp, kiểu thông điệp handshake đều có giá trị 12. Các client phải sử dụng kiến thức đã nắm được từ các thông điệp handshake trước (thuật toán trao đổi khoá từ bộ mã đã chọn trong thông điệp ServerHello và thuật toán ký, nếu liên quan từ thông điệp Certificate) để hiểu một thông điệp ServerKeyExchange một cách chính xác. Hình 2.5, minh hoạ một thông điệp trao đổi khoá Diffie-Hellman. Ba tham số Diffie-Hellman (p,q, Y_s) tạo ra các sáu trường đầu tiên sau chiều dài thông điệp. Mỗi tham số bao gồm chiều dài của nó, theo sau là giá trị thực.

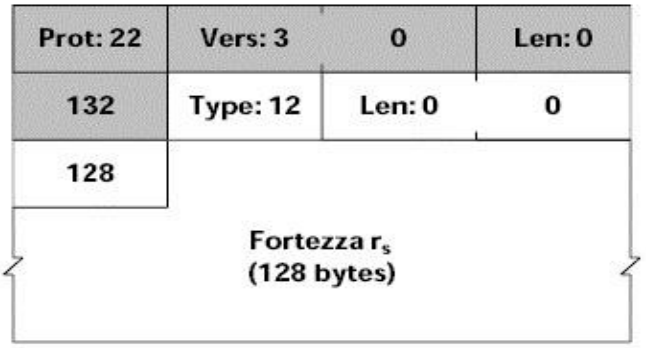


Hình 2.5. ServerKeyExchange mang các tham số Diffie-Hellman

Các thông điệp trao đổi khoá RSA được yêu cầu trong trường hợp server sử dụng RSA nhưng có một khoá RSA chỉ để ký. Client không thể gửi một khoá bí mật đã được mã hoá với khoá công khai của server. Thay vào đó, server phải tạo ra một cặp khoá public/private RSA tạm và sử dụng thông điệp ServerKeyExchange để gửi khoá công khai. Nội dung thông điệp bao gồm hai tham số của khoá công khai RSA tạm (môđun và số mũ) cộng với một ký số của các tham số đó. Mỗi tham số trong đó được lưu trong thông điệp theo định dạng: chiều dài, theo sau là giá trị.



Hình 2.6 ServerKeyExchange mang các tham số RSA



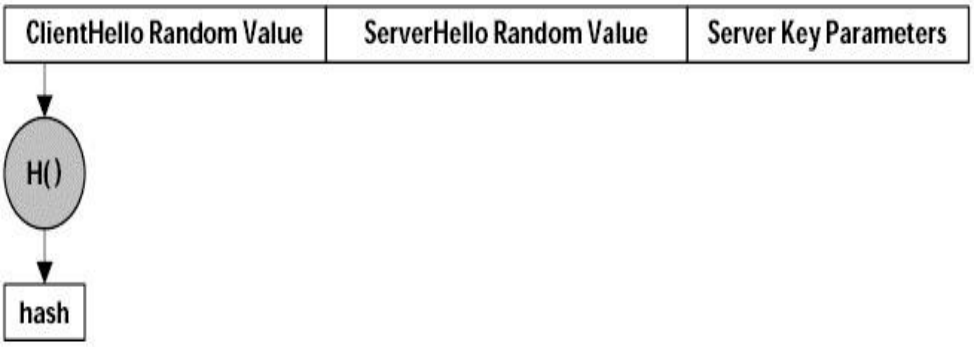
Hình 2.7. ServerKeyExchange sử dụng Fortezza

Khi các hệ thống thi hành trao đổi khoá Fortezza/DMS, thông điệp ServerKeyExchange mang giá trị Fortezza r_s, cỡ 128 byte.

ServerKeyExchange có thể bao gồm các tham số đã được ký. Định dạng chính xác của các tham số đó phụ thuộc vào thuật toán ký số mà server hỗ trợ. Nếu xác thực server không phải là một phần của một phiên SSL cụ thể thì không cần ký và thông điệp ServerKeyExchange kết thúc với các tham số Diffie-Hellman, RSA hay Fortezza.

Tuy nhiên, nếu server không ẩn danh và gửi đi một thông điệp Certificate, thì định dạng các tham số được ký tuỳ thuộc vào thuật toán ký số được thể hiện trong chứng chỉ của server. Nếu chứng chỉ của server dùng ký RSA, thì các tham số được ký bao gồm dãy hai hàm băm: một hàm băm MD5 và một hàm băm SHA. Chú ý rằng một ký số đơn cho các hàm băm kết hợp được gộp vào, không phải là các ký số riêng biệt cho

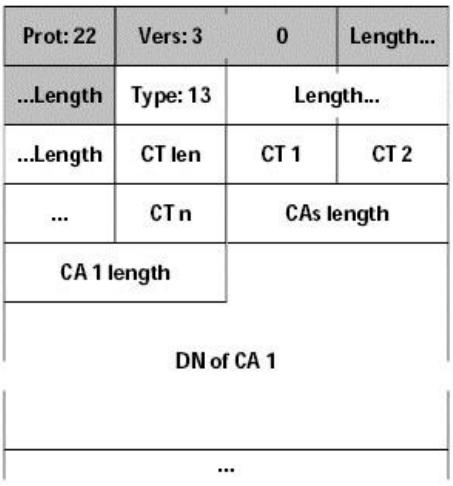
từng hàm. Nếu chứng chỉ của server là ký DSA, thì các tham số được ký chỉ bao gồm một hàm băm SHA. Trong từng trường hợp, đầu vào của các hàm băm là dữ liệu bao gồm giá trị ngẫu nhiên của client (từ ClientHello), tiếp theo là giá trị ngẫu nhiên của server (trong ServerHello), tiếp theo nữa là các tham số trao đổi khoá (các tham số Diffie-Hellman hoặc các tham số RSA). Không có tham số nào được thêm vào cho trao đổi khoá Fortezza/DMS.



Hình 2.8. Server ký một hàm băm của các tham số ServerKeyExchange

• **Thông điệp CertificateRequest**

Để xác thực định danh của client, đầu tiên server gửi một thông điệp CertificateRequest. Thông điệp này không chỉ yêu cầu client gửi các chứng chỉ của nó (và các thông tin ký sử dụng khoá riêng cho chứng chỉ đó), nó cũng đưa ra cho client các chứng chỉ mà server có thể chấp nhận.



Hình 2.9. Thông điệp CertificateRequest

Thông điệp CertificateRequest là kiểu thông điệp handshake 13, sau kiểu handshake và chiều dài thông điệp bao gồm một danh sách các kiểu chứng chỉ có thể chấp nhận được. Danh sách kiểu này bắt đầu với chiều dài của từng cái (một giá trị

một byte), sau đó là một hay nhiều giá trị byte đơn định danh kiểu chứng chỉ cụ thể. Bảng dưới đây liệt kê các giá trị kiểu chứng chỉ và ý nghĩa của chúng.

Giá trị	Kiểu chứng chỉ
1	Ký số và trao đổi khoá RSA
2	Ký số DSA
3	Ký số RSA với trao đổi khoá fixed Diffie-Hellman
4	Ký số DSA với trao đổi khoá fixed Diffie-Hellman
5	Ký số RSA với trao đổi khoá ephemeral Diffie-Hellman
6	Ký số DSA với trao đổi khoá ephemeral Diffie-Hellman
20	Ký số và trao đổi khoá Fortezza/DMS

Bảng 1. Giá trị các kiểu chứng chỉ

Bên cạnh các kiểu chứng chỉ, thông điệp CertificateRequest cũng thể hiện các quyền chứng chỉ mà server cho là thích hợp. Danh sách này bắt đầu với trường chiều dài 2 byte và sau đó bao gồm một hay nhiều cái tên khác nhau.

Mỗi một tên có trường chiều dài riêng, và các định danh một quyền chứng chỉ.

- **Thông điệp ServerHelloDone**

Thông điệp này được gửi bởi server báo hiệu kết thúc giai đoạn ServerHello và các thông điệp kết hợp. Sau khi gửi thông điệp này, server sẽ chờ đáp ứng của client. Thông điệp này có nghĩa là server đã hoàn thành việc gửi các thông điệp hỗ trợ trao đổi khoá, và client có thể bắt đầu giai đoạn trao đổi khoá của nó. Trước khi nhận được thông điệp ServerHelloDone, client sẽ kiểm tra rằng server đã cung cấp một chứng chỉ hợp lệ nếu được yêu cầu và kiểm tra các tham số server hello có thể chấp nhận được. Nếu tất cả đã thoả mãn, client sẽ gửi một hay nhiều thông điệp trở lại cho server.

Định dạng thông điệp ServerHelloDone rất đơn giản, kiểu thông điệp Handshake là 14 và chiều dài thông điệp là 0 (vì nó không mang bất kỳ thông tin nào cả).

Prot: 22	Vers: 3	0	Len: 0
4	Type: 14	Len: 0	0
0			

Hình 2.10. Thông điệp ServerHelloDone

c Giai đoạn 3: Xác thực Client và trao đổi khoá

Nếu server đã gửi một thông điệp CertificateRequest, client phải gửi thông điệp chứng chỉ của mình, sau đó là thông điệp ClientKeyExchange. Nội dung của thông điệp đó tuỳ thuộc vào thuật toán khoá công khai được lựa chọn trong thoả thuận ClientHello và ServerHello. Nếu client đã gửi một chứng chỉ với các thuộc tính ký, một thông điệp CertificateVerify được gửi để xác thực lại chứng chỉ.

• **Thông điệp ClientCertificate**

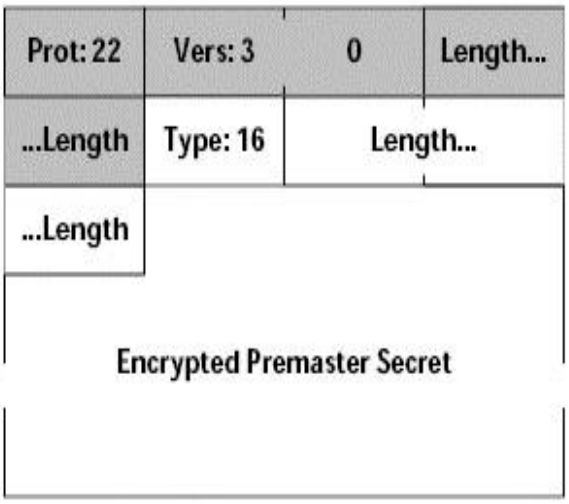
Đây là thông điệp đầu tiên client có thể gửi cho sau khi nhận được thông điệp ServerHelloDone. Thông điệp này được gửi chỉ khi server yêu cầu. Nếu không có chứng chỉ phù hợp, client sẽ gửi một thông điệp không chứa chứng chỉ nào. Nếu xác thực client được yêu cầu bởi server để quá trình bắt tay được tiếp tục, nó có thể đáp ứng lại với một cảnh báo lỗi bắt tay. Loại thông điệp tương tự và cấu trúc sẽ được sử dụng cho các đáp ứng của client tới một thông điệp ClientRequest. Chú ý rằng, một client có thể không gửi chứng chỉ nếu nó không có chứng chỉ phù hợp để gửi đáp ứng tới yêu cầu xác thực của server. Các chứng chỉ Diffie-Hellman của client phải phù hợp với các tham số Diffie-Hellman cụ thể của server.

• **Thông điệp ClientKeyExchange**

Thông điệp này luôn được gửi bởi client, theo ngay sau thông điệp ClientCertificate (nếu nó được gửi). Hay nói cách khác, nó là thông điệp đầu tiên client gửi đi sau khi nhận được thông điệp ServerHelloDone. Với thông điệp này, client cung cấp cho server các nguyên liệu khoá cần thiết cho bảo mật truyền thông; định dạng chính xác của thông điệp tuỳ thuộc vào thuật toán trao đổi khoá cụ thể mà các bên sử dụng. Ba khả năng có thể mà SSL cho phép là trao đổi khoá RSA, Diffie-Hellman, và Fortezza/DMS.

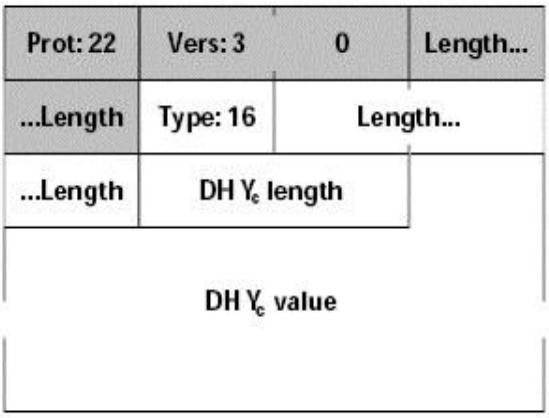
Định dạng thông điệp đầu tiên là trao đổi khoá RSA. Thông điệp có loại thông điệp handshake là 16, và chiều dài thông điệp handshake chuẩn. Phần thân thông điệp bao gồm duy nhất **premaster secret** đã mã hoá. **Premaster secret** được mã hoá sử

dụng khoá công khai của server, đã nhận được trong ServerKeyExchange hay thông điệp Certificate.



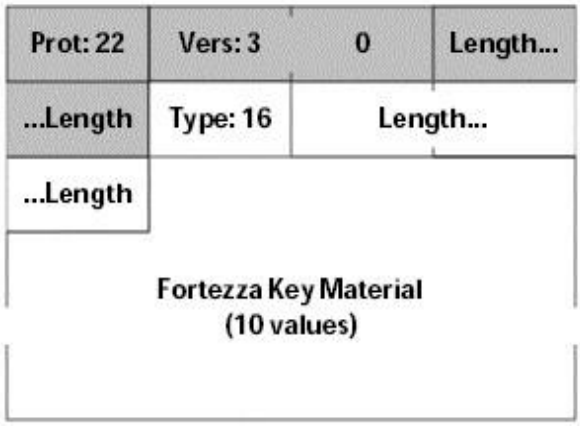
Hình 2.11. Thông điệp ClientKeyExchange với RSA

Với trao đổi khoá RSA, **premaster secret** đơn giản là hai byte lưu phiên bản SSL mà client hỗ trợ (3 và 0 cho phiên bản 3.0) tiếp theo là 46 byte ngẫu nhiên được sinh ra an toàn.



Hình 2.12. Thông điệp ClientKeyExchange với Diffie-Hellman

Với giao thức trao đổi khoá là Diffie-Hellman, có hai trạng thái có thể của thông điệp ClientKeyExchange. Nếu trao đổi Diffie-Hellman là ngắn, thì thông điệp có định dạng như hình 22. Phần thân thông điệp bao gồm giá trị Y_c của client, xếp theo chiều dài giá trị. Nếu trao đổi Diffie-Hellman là chi tiết, giá trị Y_c được lưu trong chứng chỉ của client, khi đó ClientKeyExchange sẽ trống.

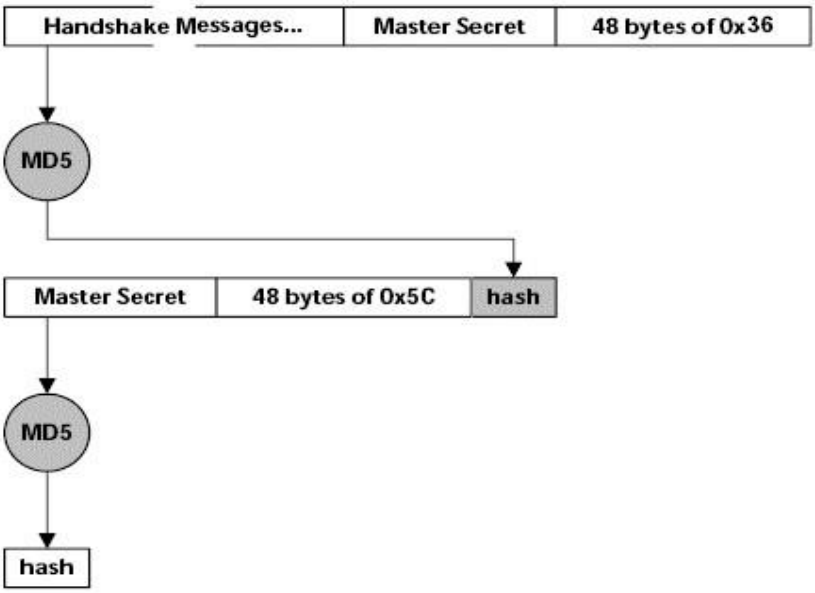


Hình 2.13. Thông điệp ClientKeyExchange với Fortezza

Với trao đổi khoá Fortezza/DMS, thông điệp ClientKeyExchange yêu cầu một tập hợp các tham số (10 giá trị).

- **Thông điệp CertificateVerify**

Thông điệp này được sử dụng để cung cấp một sự xác thực rõ ràng hơn về chứng chỉ của client. Thông điệp chỉ được gửi khi bất kỳ một chứng chỉ client nào có khả năng ký số. Khi được gửi, nó sẽ theo ngay sau thông điệp ClientKeyExchange. Thông điệp này ký trên một hàm băm mã dựa trên các thông điệp, và cấu trúc của nó được định nghĩa như sau:



Hình 2.14. Tạo thông điệp CertificateVerify

Định dạng chính xác của thông tin tuỳ thuộc vào việc chứng chỉ của client dùng ký RSA hay DSA. Với các chứng chỉ RSA, hai hàm băm riêng biệt được kết hợp lại

và ký: một hàm băm MD5 và một hàm băm SHA. Với các chứng chỉ DSA, chỉ một hàm băm SHA được tạo và được ký.

Trong mọi trường hợp, thông tin mà server đưa vào các hàm băm là như nhau. Client xây dựng thông tin trong ba bước. Đầu tiên, họ tính một giá trị đặc biệt gọi là **master secret**. Để tính giá trị của **master secret**, client tính toán theo tiến trình sau:

1. Bắt đầu với 48 byte **premaster secret**. Client tạo ra giá trị này và gửi nó tới server trong thông điệp ClientKeyExchange.
2. Tính toán hàm băm SHA của ký tự ASCII 'A' theo sau **premaster secret**, giá trị ngẫu nhiên của client (từ ClientHello) và giá trị ngẫu nhiên của server (từ ServerHello).
3. Tính hàm băm MD5 của **premaster secret** và đầu ra của bước 2.
4. Tính hàm băm SHA của hai ký tự ASCII 'BB', **premaster secret**, giá trị ngẫu nhiên của client, giá trị ngẫu nhiên của server.
5. Tính hàm băm MD5 của premaster secret và đầu ra của bước 4.
6. Kết hợp kết quả bước 3 và bước 5.
7. Tính hàm băm SHA của ba kí tự ASC 'CCC', premaster secret, giá trị ngẫu nhiên của client, giá trị ngẫu nhiên của server.
8. Tính hàm băm MD5 của premaster secret và đầu ra của bước 7.
9. Kết hợp kết quả bước 8 và bước 6

Một khi client đã có giá trị **master secret**, nó chuyển tới giai đoạn tiếp theo là tạo một hàm băm nội dung đầy đủ của các thông điệp handshake SSL trước đó đã được trao đổi trong suốt một phiên, theo sau là **master secret**, và tiếp theo nữa là giá trị byte đơn 001100110, được lặp lại 48 lần cho MD5 và 40 lần cho SHA. Trong bước thứ ba, client tạo một hàm băm mới sử dụng **master secret** tương tự, theo sau là giá trị nhị phân 01011100, được lặp lại 48 lần cho MD5, và 40 lần cho SHA, cuối cùng là đầu ra của hàm băm.

```
master secret = MD5(premaster secret + SHA('A' + premaster secret +
                        ClientHello.random + ServerHello.random))
                +
                MD5(premaster secret + SHA('BB' + premaster secret +
                        ClientHello.random + ServerHello.random))
                +
                MD5(premaster secret + SHA('CCC' + premaster secret +
                        ClientHello.random + ServerHello.random))
```

d Giai đoạn 4: Kết thúc kết nối bảo mật

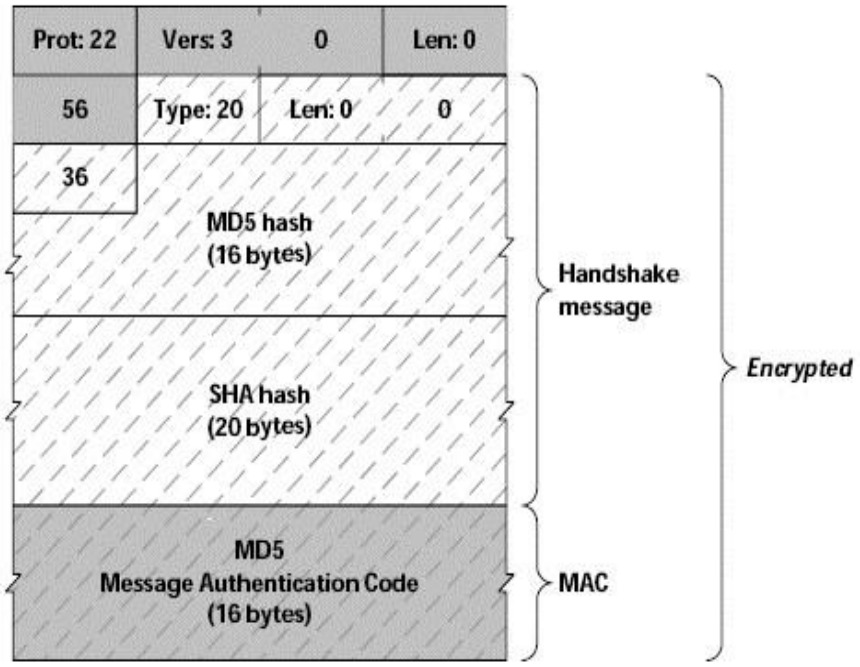
Lúc này, một thông điệp ChangeCipherSpec được gửi bởi client, và client sao chép lại CipherSpec trạng thái chờ vào CipherSpec hiện tại. Sau đó client gửi thông điệp kết thúc dưới các thuật toán mới, các khoá. Để đáp ứng lại, server sẽ gửi thông điệp ChangeCipherSpec của mình, chuyển CipherSpec chờ thành CipherSpec hiện tại, sau đó gửi thông điệp kết thúc dưới CipherSpec mới. Vào lúc này, giai đoạn bắt tay kết thúc và client và server có thể bắt đầu trao đổi dữ liệu tầng ứng dụng.

• Thông điệp ChangeCipherSpec

Client gửi một thông điệp ChangeCipherSpec và sao chép trạng thái CipherSpec chờ vào CipherSpec hiện tại. Thông điệp này được gửi ngay sau thông điệp CertificateVerify. Nó thể hiện rằng một thông điệp ChangeCipherSpec được nhận giữa các thông điệp handshake và các thông điệp kết thúc. Sẽ xảy ra một lỗi nếu thông điệp ChangeCipherSpec không được theo sau bởi thông điệp Finished.

• Thông điệp Finished

Thông điệp này luôn được gửi ngay sau thông điệp CipherSpec để kiểm tra lại các tiến trình xác thực và trao đổi khoá đã thành công. Kiểu thông điệp handshake kết thúc này là 20. Thông điệp Finish bản thân nó được mã hoá sử dụng các tham số cipher suite.

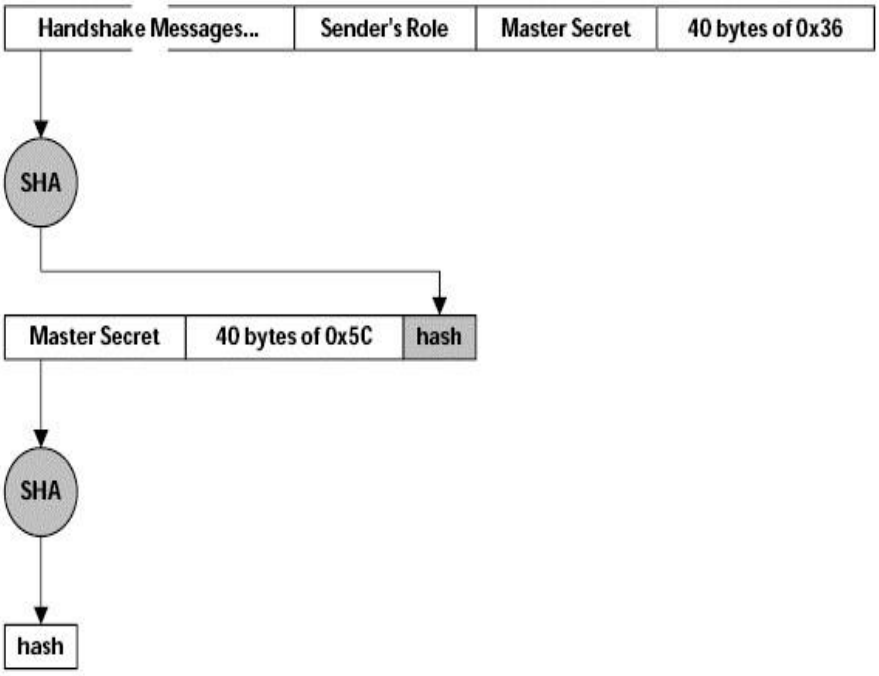


Hình 2.15. Thông điệp Finished

Phần thân thông điệp Finish bao gồm kết quả của hai hàm băm, một sử dụng thuật toán băm MD5, và một sử dụng SHA. Cả hai sự tính toán này đều sử dụng đầu vào như nhau, và đều tính trong hai bước.

Đầu tiên người gửi tạo một hàm băm nội dung đầy đủ các thông điệp handshake SSL đã trao đổi trước đó trong suốt một phiên, theo sau là một quy tắc của người gửi, master secret và padding. Quy tắc của người gửi là một giá trị hexa 434c4e54 nếu người gửi là client, 53525652 nếu là server. Padding là giá trị nhị phân 001100110, lặp lại 48 lần cho MD5, 40 lần cho SHA.

Bước thứ hai, người gửi tạo ra một hàm băm mới sử dụng master secret, theo sau là một padding thay đổi và đầu ra của hàm băm trước. Padding bước hai là một giá trị nhị phân 01011100, lặp lại 48 lần cho MD5 và 40 lần cho SHA.



Hình 2.16. Thông điệp Finished bao gồm một hàm băm

Chú ý rằng, có một sự giống nhau giữa tính toán này và tính toán hàm băm cho thông điệp CertificateVerify. Tuy nhiên cũng có hai điểm khác biệt. Đầu tiên, hàm băm Finished bao gồm quy tắc của người gửi trong khi CertificateVerify thì không. Thứ hai, tập hợp các thông điệp handshake sẽ khác khi hai hàm băm được tính. Trong mỗi trường hợp, chú ý rằng SSL không đề cập tới ChangeCipherSpec, vì thế nội dung của nó không được tính tới trong hàm băm.

Sau khi thông điệp Finished của server được gửi đi, quá trình bắt tay hoàn thành và một phiên SSL được thiết lập. Từ lúc này trở đi, mọi truyền thông giữa hai bên đều được mã hoá cho tới khi kết thúc một phiên.

2.1.2 Giao thức SSL Change Cipher Spec Protocol:

Giao thức SSL Change Cipher Spec là giao thức đơn giản nhất trong ba giao thức đặc trưng của SSL.

Giao thức này bao gồm một message đơn 1 byte giá trị là 1. Mục đích chính của message này là sinh ra trạng thái tiếp theo để gán vào trạng thái hiện tại, và trạng thái hiện tại cập nhật lại bộ mã hóa để sử dụng trên kết nối này.

SSL ChangeCipherSpec Protocol được sử dụng để thay đổi giữa một thông số mật mã này và một thông số mật mã khác.Mặc dù thông số mật mã thường được thay đổi ở cuối một sự thiết lập quan hệ SSL,nhưng nó cũng có thể được thay đổi vào bất kỳ thời điểm sau đó.

2.1.3 Giao thức SSL Alert:

Giao thức SSL Alert được dùng để truyền cảnh báo liên kết SSL với đầu cuối bên kia. Hay chính là việc nó đưa ra những thông báo mang tính ràng buộc trong quá trình giao tiếp giữa web brower và web server.

Các cảnh cáo:

- bad_record_mac: MAC không chính xác
- unsupported_certificate: dạng certificate nhận được thì không hỗ trợ.
- certificate_revoked: certificate đã bị thu hồi bởi nhà cung cấp.
- certificate_expired: certificate đã hết hạn đăng ký.

SSL Alert Protocol được sử dụng để chuyển các cảnh báo thông qua SSL Record Protocol. Mỗi cảnh báo gồm 2 phần, một mức cảnh báo và một mô tả cảnh báo.

2.1.4 SSL Record Layer:

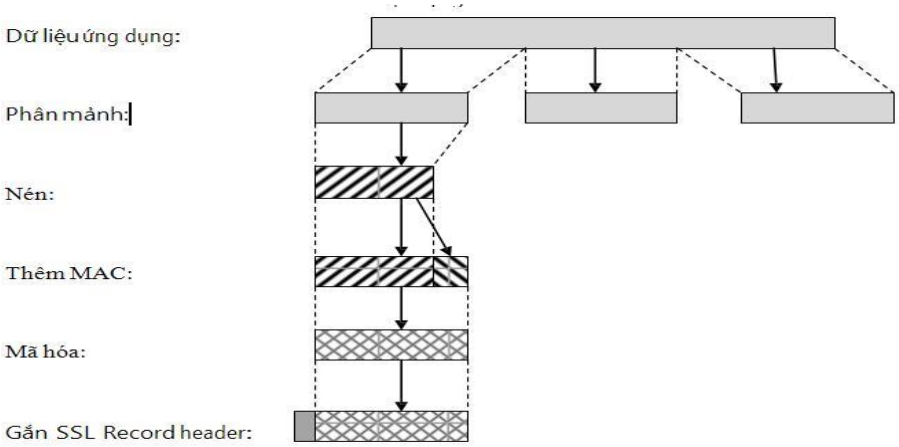
SSL Record Layer xác định khuôn dạng cho tiến hành mã hóa và truyền tin hai chiều giữa hai đối tượng.

SSL Record Protocol cung cấp 2 dịch vụ cho kết nối SSL:

- ✓ Tính bảo mật (Confidentiality): Handshake Protocol định nghĩa 1 khóa bí mật được chia sẻ, khóa này được sử dụng cho mã hóa quy ước các dữ liệu SSL

- ✓ Tính toàn vẹn thông điệp (Message integrity): Handshake Protocol cũng định nghĩa 1 khóa bí mật được chia sẻ, khóa này được sử dụng để hình thành MAC (mã xác thực message).

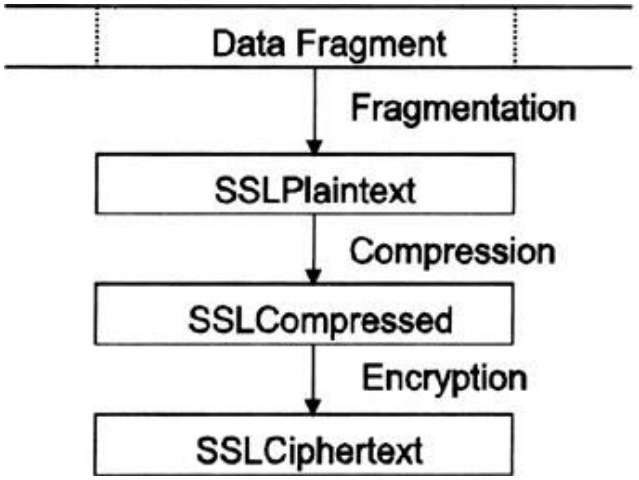
Toàn bộ hoạt động của SSL Record Protocol:



Hình 2.17. Toàn bộ hoạt động của SSL Record Protocol

SSL Record Protocol nhận dữ liệu từ các giao thức con SSL lớp cao hơn và xử lý việc phân đoạn, nén, xác thực và mã hóa dữ liệu. Chính xác hơn, giao thức này lấy một khối dữ liệu có kích cỡ tùy ý làm dữ liệu nhập và tạo một loạt các bản ghi nhỏ hơn hoặc bằng 16,383 byte.

Các bước SSL Record Protocol:

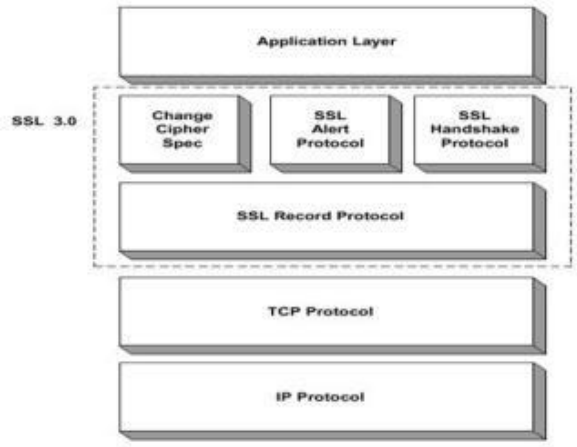


Hình 2.18. Các bước SSL Record Protocol

Các bước khác nhau của SSL Record Protocol vốn đi từ một đoạn dữ liệu thô đến một bản ghi SSL Plaintext (bước phân đoạn), SSL Compressed (bước nén) và SSL Ciphertext (bước mã hóa) được minh họa trong hình trên.

Sau cùng, mỗi bản ghi SSL chứa các trường thông tin sau đây:

- Loại nội dung
- Số phiên bản của giao thức
- Chiều dài
- Tải trọng dữ liệu (được nén và được mã hóa tùy ý)
- MAC

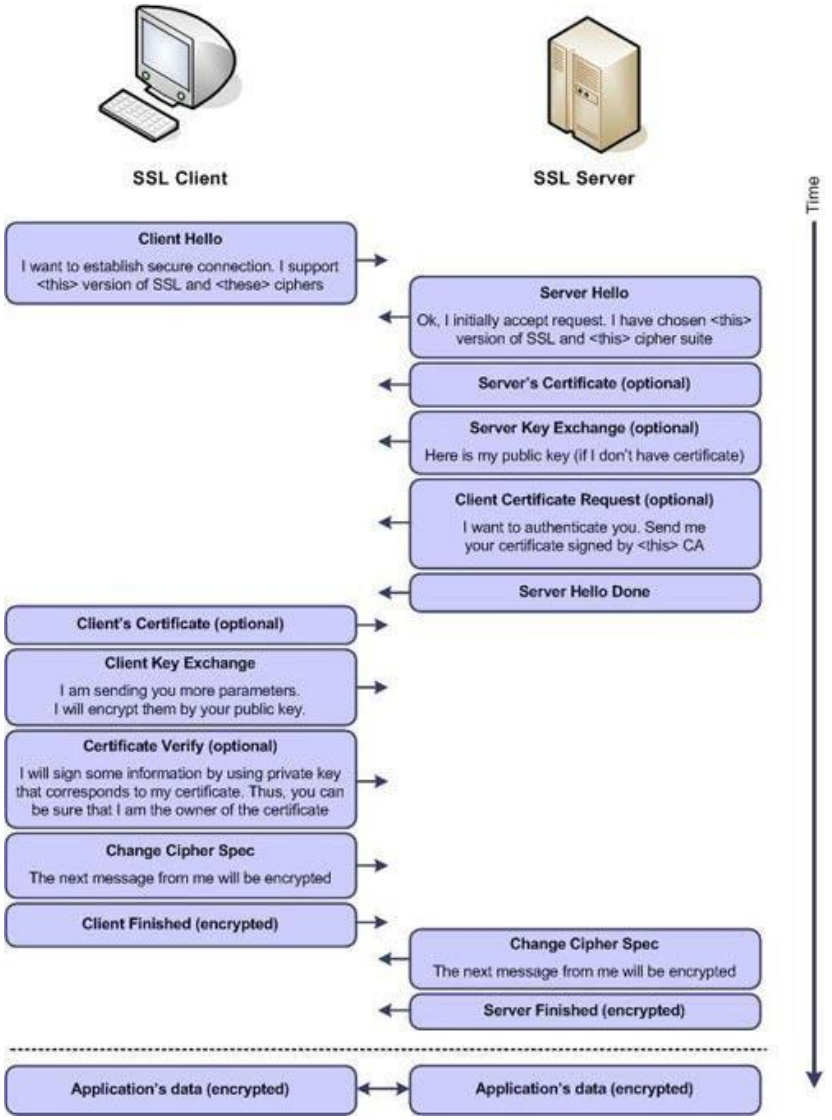


Hình 2.19. Cấu trúc mẩu tin SSL Record Protocol

Loại nội dung xác định giao thức lớp cao hơn vốn phải được sử dụng để sau đó xử lý tải trọng dữ liệu bản ghi SSL (sau khi giải nén và giải mã hóa thích hợp). Số phiên bản của giao thức xác định phiên bản SSL đang sử dụng (thường là version 3.0).

Mỗi tải trọng dữ liệu bản ghi SSL được nén và được mã hóa theo phương thức nén hiện hành và thông số mật mã được xác định cho session SSL. Lúc bắt đầu mỗi session SSL, phương pháp nén và thông số mật mã thường được xác định là rỗng. Cả hai được xác lập trong suốt quá trình thực thi ban đầu SSL Handshake Protocol. Sau cùng, MAC được thêm vào mỗi bản ghi SSL. Nó cung cấp các dịch vụ xác thực nguồn gốc thông báo và tính toàn vẹn dữ liệu. Tương tự như thuật toán mã hóa, thuật toán vốn được sử dụng để tính và xác nhận MAC được xác định trong thông số mật mã của trạng thái session hiện hành. Theo mặc định, SSL Record Protocol sử dụng một cấu trúc MAC vốn tương tự nhưng vẫn khác với cấu trúc HMAC.

2.2 Hoạt Động Của Giao Thức SSL



Hình 2.20. Khái quát cơ chế hoạt động của giao thức SSL

Từng bước thành lập một kết nối SSL:

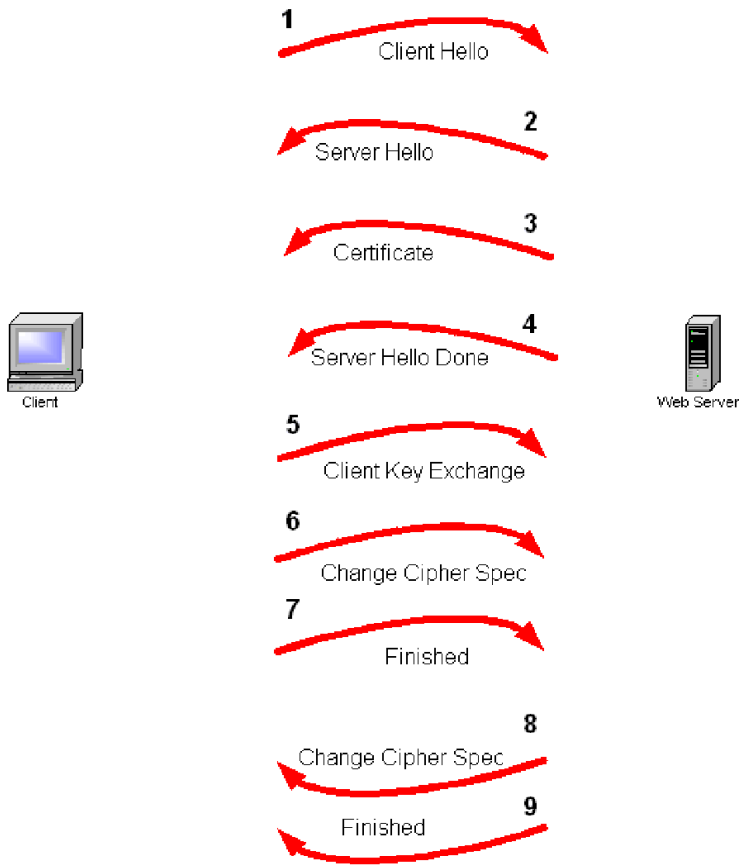
Khi hai ứng dụng máy tính, thí dụ giữa một trình duyệt web và máy chủ web, làm việc với nhau, máy chủ và máy khách sẽ trao đổi "lời chào" (hellos) dưới dạng các thông điệp cho nhau với xuất phát đầu tiên chủ động từ máy chủ, đồng thời xác định các chuẩn về thuật toán mã hoá và nén số liệu có thể được áp dụng giữa hai ứng dụng. Ngoài ra, các ứng dụng còn trao đổi "số nhận dạng/khoá theo phiên" (session ID, session key) duy nhất

cho lần làm việc đó. Sau đó ứng dụng khách (trình duyệt) yêu cầu có chứng thực điện tử (digital certificate) xác thực của ứng dụng chủ (web server).

Chứng thực điện tử thường được xác nhận rộng rãi bởi một cơ quan trung gian (là CA - Certificate Authority) như RSA Data Sercurity hay VeriSign Inc... một dạng tổ chức độc lập, trung lập và có uy tín. Các tổ chức này cung cấp dịch vụ "xác nhận" số nhận dạng của một công ty và phát hành chứng chỉ duy nhất cho công ty đó như là bằng chứng nhận dạng (identity) cho các giao dịch trên mạng, ở đây là các máy chủ webserver.

Sau khi kiểm tra chứng chỉ điện tử của máy chủ (sử dụng thuật toán mật mã công khai, như RSA tại trình máy trạm), ứng dụng máy trạm sử dụng các thông tin trong chứng chỉ điện tử để mã hoá thông điệp gửi lại máy chủ mà chỉ có máy chủ đó có thể giải mã. Trên cơ sở đó, hai ứng dụng trao đổi khoá chính (master key) - khoá bí mật hay khoá đối xứng - để làm cơ sở cho việc mã hoá luồng thông tin/dữ liệu qua lại giữa hai ứng dụng chủ khách.

2.3 Cơ chế hoạt động SSL



Hình 2.21. Cách thức làm việc SSL

1. Bước đầu tiên trong quá trình này là dành cho các khách hàng để gửi một máy chủ tin nhắn "Hello". Thông điệp này chứa phiên bản bộ mã hóa SSL và khách hàng có thể nói chuyện. Khách hàng sẽ gửi chi tiết tối đa chiều dài chính của nó vào lúc này.

2. Máy chủ trả về tin nhắn "Hello" cho khách hàng, trong đó nó chỉ định phiên bản của SSL và các thuật toán mật mã và độ dài khóa được sử dụng trong cuộc đàm thoại,được lựa chọn từ sự lựa chọn được cung cấp trong các khách hàng chào.

3. Máy chủ sẽ gửi giấy chứng nhận kỹ thuật số cho khách hàng để kiểm tra. Hầu hết các trình duyệt hiện đại tự động kiểm tra giấy chứng nhận (phụ thuộc vào cấu hình) và cảnh báo người dùng nếu đó là không hợp lệ.

4. Máy chủ gửi một thông điệp máy chủ thực hiện lưu ý nó đã kết luận ban đầu của trình tự thiết lập.

5. Khách hàng tạo ra một khóa đối xứng và mã hóa nó bằng cách sử dụng khóa công khai của máy chủ (cert). Sau đó nó sẽ gửi thông điệp này đến máy chủ.

6. Khách hàng sẽ gửi tin nhắn "Change Cipher Spec" mật mã cho máy chủ tất cả các thông tin liên lạc trong tương lai nên được bằng khóa mới.

7. Khách hàng bây giờ gửi một thông điệp "Finised" bằng cách sử dụng các phím mới để xác định nếu máy chủ là có thể giải mã tin nhắn và đàm phán thành công.

8. Máy chủ gửi một thông điệp "Change Cipher Spec" nói với khách hàng rằng tất cả các thông tin liên lạc trong tương lai sẽ được mã hóa.

9. Máy chủ gửi thông điệp "Finished" riêng của mình được mã hóa bằng cách sử dụng phím. Nếu khách hàng có thể đọc tin nhắn này sau đó đàm phán kết thúc thành công.

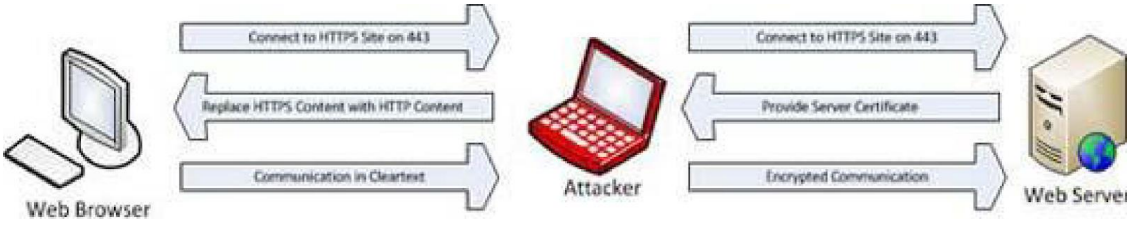
2.4 Tấn công và cách phòng chống

2.4.1 Tấn công

- Quá trình tấn công (Attack): trong hầu hết các trường hợp, SSL chưa bao giờ bị trực tiếp tấn công. Hầu hết thời gian một kết nối SSL được khởi tạo thông qua HTTPS.

- Ý tưởng: Nếu bạn tấn công một phiên giao dịch từ một kết nối không an toàn đến một kết nối an toàn, trong trường hợp này là từ HTTP vào HTTPS, bạn sẽ tấn công cầu nối và có thể "Man-in-the-middle" kết nối SSL trước khi nó xuất hiện. Để thực hiện hiệu quả điều này, Moxie đã tạo một công cụ SSLstrip, chúng ta sẽ sử dụng công cụ này dưới đây.

Chiếm quyền điều khiển truyền thông HTTPS:



Lưu lượng giữa máy khách và máy chủ đầu tiên sẽ bị chặn.

Khi bắt gặp một HTTPS URL, sslstrip sẽ thay thế nó bằng một liên kết HTTP và sẽ ánh xạ những thay đổi của nó.

Máy tấn công sẽ cung cấp các chứng chỉ cho máy chủ web và giả mạo máy khách.

Lưu lượng được nhận trở lại từ website an toàn và được cung cấp trở lại cho máy khách.

Quá trình làm việc khá tốt, máy chủ có liên quan vẫn nhận lưu lượng SSL mà không hề biết về sự khác biệt này. Chỉ có một sự khác biệt rõ rệt trong trải nghiệm người dùng là lưu lượng sẽ không được cắm cờ HTTPS trong trình duyệt, vì vậy một người dùng có kinh nghiệm sẽ có thể thấy đó là một điều dị thường.

Xem xét quá trình tấn công sử dụng SSL-Trip:

Trước tiên, phân phối Linux mà bạn đang sử dụng phải được cấu hình để chuyển tiếp IP.

Để thực hiện điều này, hãy nhập lệnh echo"1">/proc/sys/net/ipv4/ip_forward vào một shell.

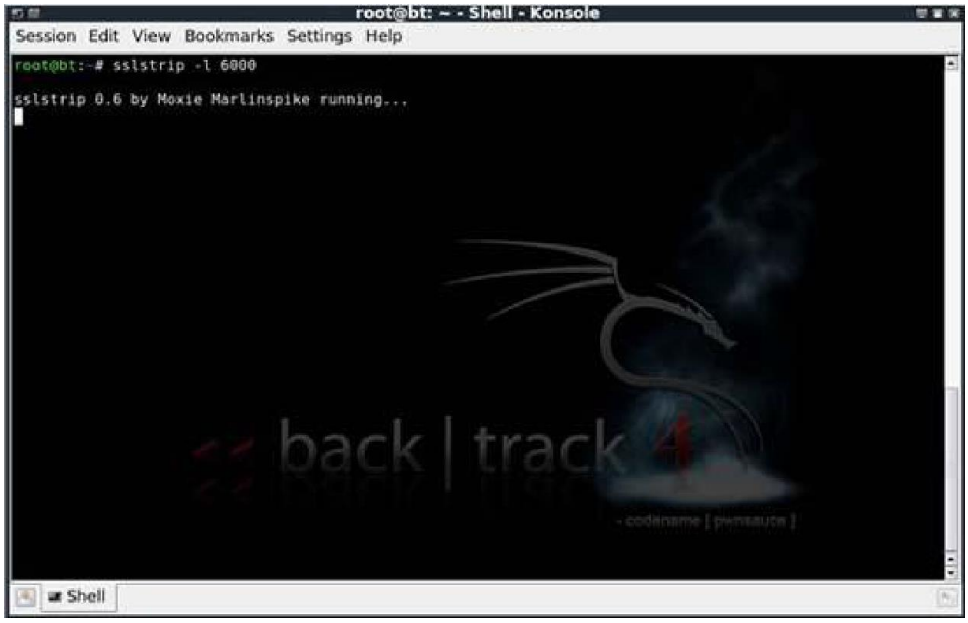
Khi thực hiện xong, chúng ta phải làm cho tất cả lưu lượng HTTP được chặn sẽ được định tuyến đến cổng mà ở đó SSLstrip sẽ lắng nghe. Điều này được thực hiện bằng cách thay đổi cấu hình iptables của tường lửa.

Sử dụng lệnh:

iptables -t nat -A PREROUTING -p tcp --destination-port 80 -j REDIRECT --to-port <listenPort>.

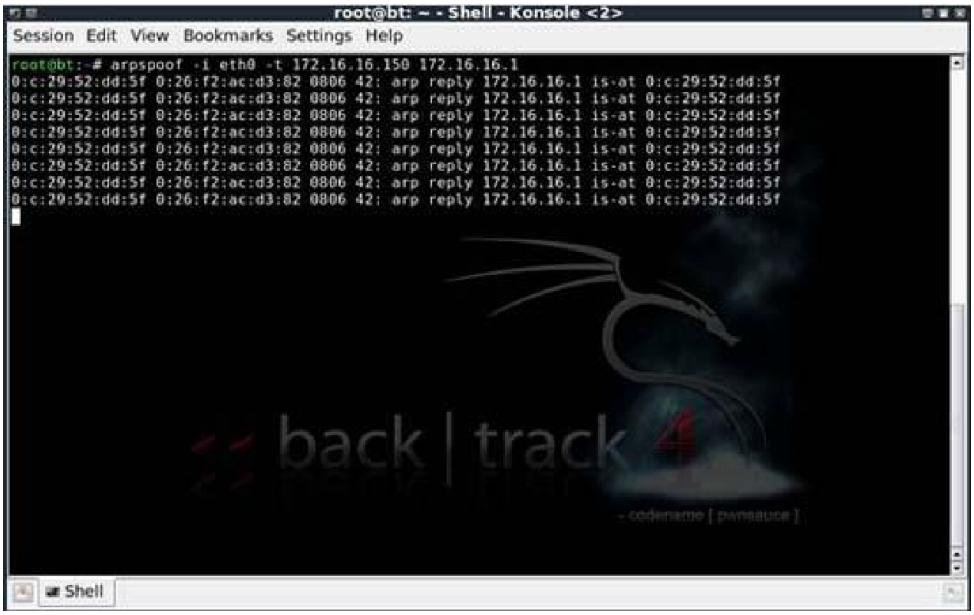
Thay thế <listenPort> bằng một cổng nào đó theo lựa chọn của mình. Sau khi thực hiện xong việc cấu hình này, chúng ta có thể chạy sslstrip và cấu hình sao cho nó có thể lắng nghe trên cổng được chỉ định bằng lệnh:

sslstrip -l <listenPort>.



Bước cuối cùng trong quá trình này là cấu hình giả mạo ARP để chặn lưu lượng của host đích. Chúng ta sẽ sử dụng tiện ích arpspoof có trong Backtrack 4.

Lệnh để thực hiện là arpspoof -i <interface> -t <targetIP> <gatewayIP>.



2.4.2 Biện pháp phòng chống

Sử dụng kết nối an toàn HTTPS: Khi bạn thực hiện tấn công được mô tả ở đây, nó sẽ lấy đi khía cạnh an toàn của kết nối, thứ có thể xác định được trong trình duyệt. Điều này có nghĩa rằng nếu bạn đăng nhập vào tài khoản ngân hàng trực tuyến và thấy rằng nó chỉ là một kết nối HTTP chuẩn thì chắc chắn có thứ gì đó sai ở đây. Bất cứ khi nào trình duyệt

mà bạn chọn sử dụng cũng cần bảo đảm rằng bạn biết cách phân biệt các kết nối an toàn với những kết nối không an toàn.

Lưu tài khoản ngân hàng trực tuyến ở nhà cơ hội cho ai đó có thể chặn lưu lượng của bạn trên mạng gia đình sẽ ít hơn nhiều so với mạng ở nơi làm việc của bạn. Điều này không phải vì máy tính gia đình của bạn an toàn hơn (mà sự thật có lẽ là kém an toàn hơn), nhưng sự thật nếu chỉ có một hoặc hai máy tính ở nhà thì các bạn chỉ phải quan tâm đến việc chiếm quyền điều khiển phiên nếu có ai đó trong nhà bạn bắt đầu xem và tập theo các video về hacking trên YouTube.

Bảo mật các máy tính bên trong mạng: Các tấn công thường được thực thi bên trong một mạng. Nếu các thiết bị mạng của bạn được an toàn thì nguy cơ bị thỏa hiệp các host để sau đó được sử dụng để khởi chạy tấn công chiếm quyền điều khiển phiên cũng sẽ giảm.

2.5 Kết luận chương

Chương 2, nghiên cứu sâu về cách thức hoạt động, các mô hình tấn công cơ bản trên giao thức SSL. Hiểu được quá trình bắt tay (handshake) để tạo kết nối an toàn giữa client và server và ghi lại (record) để cung cấp tính bảo mật và toàn vẹn dữ liệu cho việc truyền các gói tin.

CHƯƠNG 3. ĐÁNH GIÁ GIAO THỨC SSL

3.1 Đặc tính giao thức SSL

Giao thức SSL cung cấp sự bảo mật truyền thông có ba đặc tính cơ bản:

- Các bên giao tiếp (nghĩa là client và server) có thể xác thực nhau bằng cách sử dụng mật mã khóa chung.
- Sự bí mật của lưu lượng dữ liệu được bảo vệ vì nối kết được mã hóa trong suốt sau khi một sự thiết lập quan hệ ban đầu và sự thương lượng khóa session đã xảy ra.
- Tính xác thực và tính toàn vẹn của lưu lượng dữ liệu cũng được bảo vệ vì các thông báo được xác thực và được kiểm tra tính toàn vẹn một cách trong suốt nhờ bằng cách sử dụng MAC.

3.2 Ưu điểm của SSL

Tính năng mạnh nhất của SSL/TLS là chúng xác định mối quan hệ với các tầng giao thức khác như thế nào trong hệ thống kiến trúc mạng OSI. Tại mức cao nhất là phần mềm ứng dụng hoặc các trình duyệt. Chạy phía dưới các ứng dụng này là giao thức tầng ứng dụng bao gồm Telnet, FTP, HTTP…Bên dưới nữa là giao thức SSL và các thuật toán mã hoá được sử dụng để kết nối. Bên dưới SSL là tầng giao vận. Hầu hết các trường hợp đó là TCP/IP. Tuy nhiên, giao thức SSL là duy nhất, không phụ thuộc vào giao thức mạng. Bởi vì SSL không phụ thuộc vào các tầng giao thức cho nên SSL trở thành một nền tảng độc lập hay là một thực thể mạng độc lập.

Một ưu điểm khác của SSL đó là ngăn chặn cách thức tấn công từ điển. nhiên được Server sử dụng, nonce number là một số không thể bị phá khoá.

Giao thức SSL còn bảo vệ chính nó với đối tác thứ 3. Đó là các Client xâm nhập bất hợp pháp dữ liệu trên đường truyền. Client xâm nhập này có thể giả mạo Client hoặc Server, SSL ngăn chặn sự giả mạo này bằng cách sử dụng khoá riêng của Server và sử dụng chứng chỉ số.

- SSL được ứng dụng rộng rãi trong các giao dịch yêu cầu thanh toán qua mạng.

- Được hỗ trợ bởi hầu hết các trình duyệt và các phần mềm phía server.

- Được thiết kế độc lập với tầng ứng dụng nên có thể sử dụng cho nhiều ứng dụng khác nhau.

- Mọi hoạt động đều trong suốt với người sử dụng.

3.3 Giới hạn SSL

Giao thức SSL cũng như bất kỳ công nghệ nào bản thân nó cũng sẽ tồn tại những giới hạn. Và khi nó là một công cụ bảo mật thì việc hiểu những giới hạn của nó là cực kỳ quan trọng.

Giới hạn của SSL chủ yếu tập trung ở 3 điểm chính:

- Không có những cơ chế xác nhận người dùng một cách chắc chắn. Người mua hàng có nguy cơ bị lộ thông tin về tài khoản. Nhiều người dùng vẫn đang dùng SSL V2.0 thay vì SSL V3.0 và không có có cơ chế xác nhận lẫn nhau trong quá trình thiết lập giao thức bắt tay. Và nhất là việc,trên thực tế khi sử dụng giao thức này, người dùng vẫn có khả năng bị các hacker dò tìm ra khoá bí mật dùng để mã hoá thông tin .
- Giới hạn về giao thức truyền tải: Mặc dù SSL được thiết kế để thích hợp với nhiều loại ứng dụng nhưng ngay từ lúc ban đầu ý định của nhà thiết là chủ yếu tập trung vào các giao dịch trên Web. SSL là một giao thức đòi hỏi sự tin cậy cao vì thế giao thức truyền tải của nó cũng là TCP thay vì UDP và bản thân nó cũng không được hỗ trợ chạy trên UDP.
- Giới hạn về công cụ: SSL đơn giản chỉ là một giao thức liên lạc,nó hoạt động dựa vào sự hợp thành của nhiều chức năng, bao gồm nhiều loại thuật toán khác nhau. Những thuật toán này có nhiệm vụ là mã hóa và giải mã. Và SSL sẽ không thực sự mạnh mẽ nếu không có các công cụ này. Như thế có thể thấy rằng bản thân SSL không có điểm yếu mà điểm yếu của các thuật toán mã hóa mới thực sự quan trọng.
- Giới hạn về môi trường: Bất cứ một giao thức bảo mật mạng nào sinh ra hầu như đều chỉ cung cấp an toàn dữ liệu khi đi trên đường truyền. Không giao thức bảo mật mạng nào có thể bảo vệ dữ liệu trước khi và sau khi gửi nó đến đích và SSL cũng thế.

Tuy nhiên, điều quan trọng cần lưu ý là SSL không ngăn được các cuộc tấn công lưu lượng. Ví dụ: Bằng cách xem xét các địa chỉ IP nguồn và đích không được mã hoá và các số cổng TCP, hoặc xem xét lượng dữ liệu được truyền, một người phân tích lưu lượng vẫn có thể xác định các bên nào đang tương tác, các loại dịch vụ đang được sử dụng và đôi khi ngay cả dành được thông tin về các mối quan hệ doanh nghiệp hoặc cá nhân.

Hơn nữa, SSL không ngăn được các cuộc tấn công có định hướng dựa vào phần thực thi TCP. Ví dụ: như các cuộc tấn công làm tràn ngập TCP SYN chiếm đoạt session.

3.3 Kết luận chương

Chương 3 đề cập đến các đặc tính, ưu điểm và hạn chế của giao thức bảo mật SSL. Để từ đó có những hướng giải quyết phù hợp.

KẾT LUẬN

Thông qua nội dung đã trình bày ở trên, chúng ta đã tìm hiểu chi tiết về giao thức SSL. Cụ thể chương 1, khái quát lịch sử hình thành và có cái nhìn tổng quan về giao thức, kết hợp với tìm hiểu cấu trúc bản ghi của SSL bao gồm những gì. Hiểu được quá trình bắt tay (handshake) để tạo kết nối an toàn giữa client và server và ghi lại (record) để cung cấp tính bảo mật và toàn vẹn dữ liệu cho việc truyền các gói tin. Chương 2, nghiên cứu sâu về cách thức hoạt động, các mô hình tấn công cơ bản trên giao thức SSL. Chương 3 nêu ra được các đặc tính, ưu điểm và hạn chế của giao thức SSL. Để từ đó có các hướng giải quyết phù hợp.

Chức năng cốt lõi của chứng chỉ SSL là bảo vệ giao tiếp máy chủ-máy khách. Khi cài đặt SSL, mọi bit thông tin đều được mã hóa. Dữ liệu bị khóa và chỉ có thể được mở khóa bởi người nhận dự kiến (trình duyệt hoặc máy chủ) vì không ai khác có thể có chìa khóa để mở nó. Trong khi xử lý các dữ liệu nhạy cảm như ID, mật khẩu, số thẻ tín dụng, v.v., SSL giúp bạn bảo vệ khỏi đội quân tin tặc và kẻ đọc trộm tinh quái. Khi dữ liệu được SSL chuyển thành định dạng không thể giải mã được, các kỹ năng của tin tặc chứng tỏ là một con dao không lưỡi chống lại công nghệ mã hóa vượt trội của chứng chỉ SSL.

Nhiệm vụ chính thứ hai của chứng chỉ SSL là cung cấp xác thực cho một trang web. Xác minh danh tính là một trong những khía cạnh quan trọng nhất liên quan đến bảo mật web. Đây là lúc chứng chỉ SSL phát huy tác dụng.

Bất cứ một giao thức bảo mật mạng nào sinh ra hầu như đều chỉ cung cấp an toàn dữ liệu khi đi trên đường truyền.Không giao thức bảo mật mạng nào có thể bảo vệ dữ liệu trước khi và sau khi gửi nó đến đích và SSL cũng thế.

Ngày nay việc bảo mật thông tin là yếu tố quan trọng để quyết định sự sống còn của một tổ chức, một công ty hay doanh nghiệp. Với sự phát triển nhanh chóng của công nghệ đã mang lại nhiều tiện ích cho người dùng nhưng đồng thời cũng đặt ra một nhu cầu hết sức cấp thiết về sự an toàn và bảo mật. Và qua việc "tìm hiểu SSL trên" thì việc sử dụng giao thức SSL chính là giải pháp tốt nhất hiện nay đáp ứng những nhu cầu đó và nó được coi như là "lá chắn cuối cùng" trong bảo mật thương mại điện tử.

TÀI LIỆU THAM KHẢO

1 Bài giảng An ninh mạng viễn thông – TS.Nguyễn Chiến Trinh, PGS.TS.Nguyễn Tiến Ban, TS.Hoàng Trọng Minh, ThS.Nguyễn Thanh Trà, ThS.Phạm Anh Thư

2 (Computer Security) Rolf Oppliger - SSL and TLS_ Theory and Practice-Artech House Publishers (2016)

3 Stephen A. Thomas - SSL & TLS Essentials_ Securing the Web-Wiley (2000)

4 Rescorla, E., The Transport Layer Security (TLS) Protocol Version 1.3," Internet-Draft, October
2015

5 Eastlake, D., "Transport Layer Security (TLS) Extensions: Extension Definitions," Standards Track RFC 6066, January 2011

6 Brown, M., and R. Housley, "Transport Layer Security (TLS) Authorization Extensions," Experimental RFC 5878, May 2010

7 Friedl, S., et al., "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension," Standards Track RFC 7301, July 2014

8 Joshua Davies(auth.) - Implementing SSL_TLS Using Cryptography and PKI (2011)

9 Network Security with OpenSSL. O'Reilly & Associates, Inc, 2002.