


CS458/CS558

Introduction to Computer Security



Chapter 17.2


Secure Socket Layer (SSL)

http://en.wikipedia.org/wiki/Transport_Layer_Security

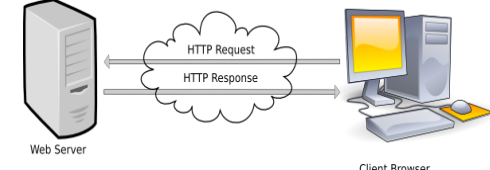


Web Security


- ◆ World Wide Web is fundamentally a **client/server application** running over internet and TCP/IP intranet.
- ◆ Web now widely used by business, government, individuals
- ◆ But Web is **vulnerable**



HTTP Protocol




- ◆ **HTTP (Hyper Text Transfer Protocol):** specifies the communication between Browsers and Web Servers.
- ◆ HTTP is a stateless protocol (server maintains no information about past client requests.)



HTTP Request

- ◆ **HTTP request**
 - ❖ HTTP header
 - Request line (e.g. `GET /images/logo.png HTTP/1.1`)
 - Request header (e.g., `Accept-Language: en`)
 -
 - ❖ HTTP body (optional)



HTTP Request: An Example

- ◆ E.g. the browser translated the URL `http://www.test101.com/doc/test.html` into the following http request:

```

post /doc/test.html HTTP/1.1
Host: www.test101.com
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 35

bookId=12345&author=Tan+Ah+Teck

```

Annotations for the example request:

- Request Line:** `post /doc/test.html HTTP/1.1`
- Request Headers:** `Host: www.test101.com`, `Accept: image/gif, image/jpeg, */*`, `Accept-Language: en-us`, `Accept-Encoding: gzip, deflate`, `User-Agent: Mozilla/4.0`, `Content-Length: 35`
- Request Message Header:** The entire header section (Request Line + Request Headers).
- Request Message Body:** `bookId=12345&author=Tan+Ah+Teck`
- A blank line separates header & body:** The empty line between the headers and the body.

HTTP Response

- HTTP Response is composed of a header and a body, separated by an empty line.
 - Header
 - Protocol version (e.g., HTTP/1.0 or HTTP/1.1)
 - Status code (e.g. 404)
 - Reason Phrase (e.g. Not Found)
 -
 - Body: a byte stream

HTTP Response: Example

```

HTTP/1.1 200 OK
Date: Sun, 08 Feb xxxx 01:11:12 GMT
Server: Apache/1.3.29 (Win32)
Last-Modified: Sat, 07 Feb xxxx
ETag: "0-23-4024c3a5"
Accept-Ranges: bytes
Content-Length: 35
Connection: close
Content-Type: text/html

<h1>My Home page</h1>

```

Diagram labels:

- Status Line: HTTP/1.1 200 OK
- Response Headers: Date, Server, Last-Modified, ETag, Accept-Ranges, Content-Length, Connection, Content-Type
- Response Message Header: (Grouping label for headers)
- A blank line separates header & body
- Response Message Body: <h1>My Home page</h1>

Hypertext Transfer Protocol Secure (Https)

- A combination of the http protocol and a network security protocol
- Also known as **Hypertext Transfer Protocol over Secure Socket Layer**
- The administrator must create a **public key certificate** for the Web server.
- This certificate must be signed by a certificate authority.
 - SSL certificate providers: Verisign, Thawte, InstantSSL, Entrust, Baltimore, Geotrust etc.
- Web browsers are distributed with the public key of major certificate authorities so that they can verify certificates signed by them.

SSL (Secure Socket Layer)

- A cryptographic protocol that provides security for communications over networks.
- One of the most widely used Web security mechanism.
- Transport layer security service** - designed to make use of TCP to provide a reliable end-to-end security service.
- Originally developed by **Netscape**
- Subsequently became Internet standard known as **TLS (Transport Layer Security)**

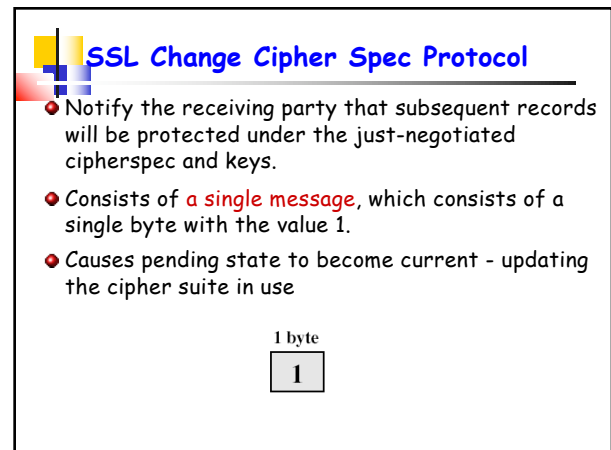
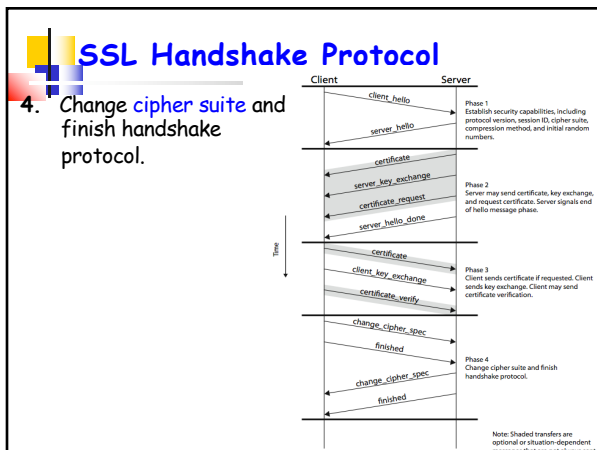
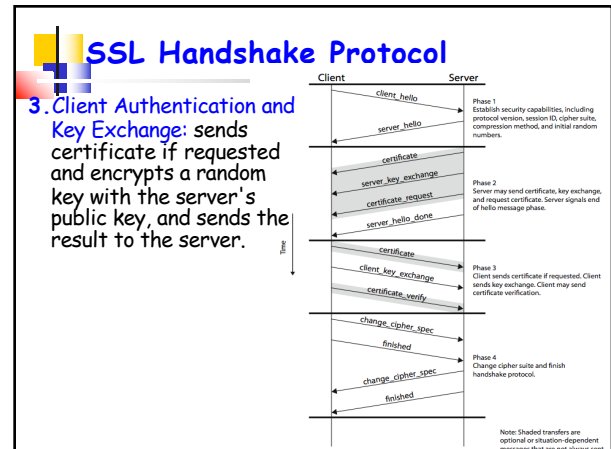
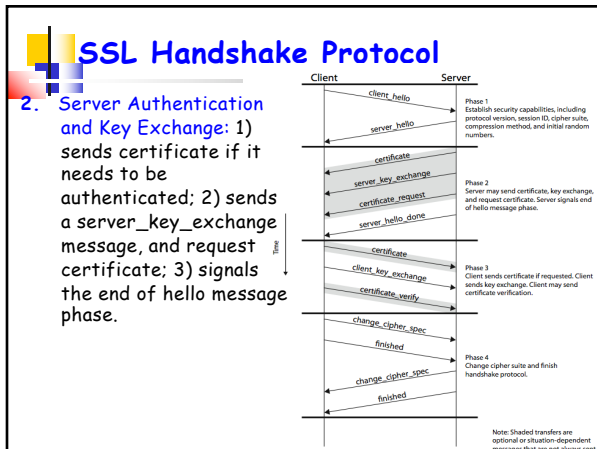
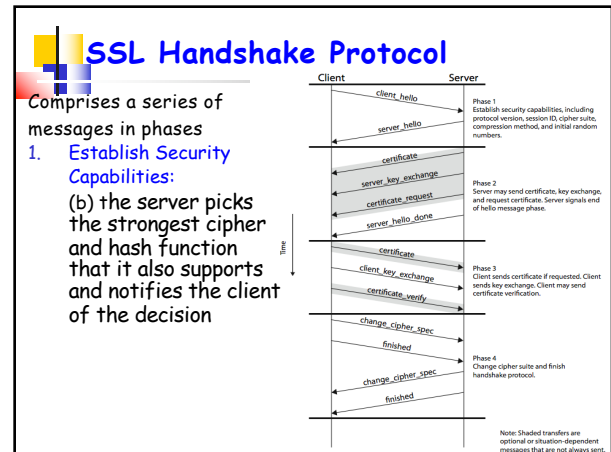
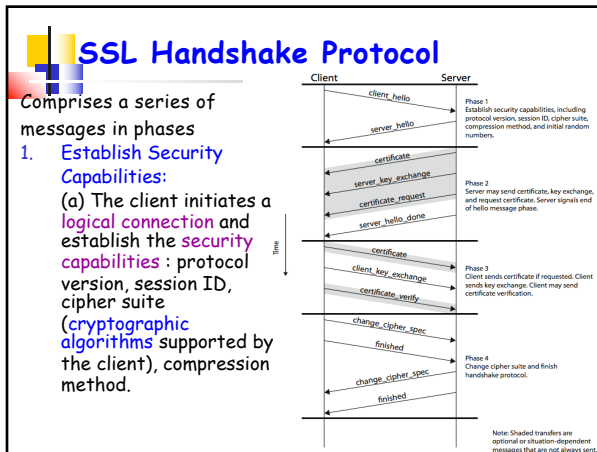
HTTP	FTP	SMTP
SSL or TLS		
TCP		
IP		

SSL Architecture

- Has two layers of protocols
 - Level 1:
 - SSL Handshake Protocol
 - SSL Change Cipher Spec Protocol
 - SSL Alert Protocol
 - HTTP
 - Level 2:
 - SSL Record Protocol
 - TCP
 - IP
- Level 1:
 - SSL Record Protocol: provides basic security services to various higher-layer protocols.
- Level 2:
 - Hypertext Transfer Protocol (HTTP): which provides the transfer service for Web client/server interaction, can operate on top of SSL.
 - Three higher-layer protocols: used in the management of SSL exchanges.

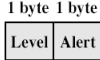
SSL Handshake Protocol

- Allows **server** and **client** to:
 - Authenticate each other
 - To negotiate encryption & MAC algorithms
 - To negotiate cryptographic keys to be used to protect data sent in an SSL record.



SSL Alert Protocol

- Conveys **SSL-related alerts** to peer entity
- Consists of two bytes - the first takes the value: warning (1) or fatal (2); the second contains a code that indicates the specific alert.



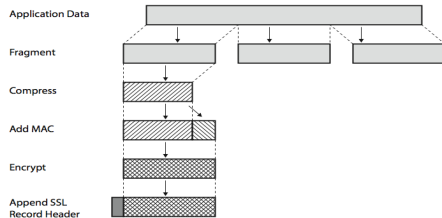
- Fatal:** unexpected message, decompression failure, handshake failure, illegal parameter
 - SSL immediately terminates the connection
- Warning:** close notify (the sender will not send any more message of this connection), bad certificate, unsupported certificate, certificate revoked, certificate expired, certificate unknown.

SSL Record Protocol Services

- Provides two services for SSL connections
 - Confidentiality:**
 - encrypt SSL payloads
 - Message integrity:**
 - use a shared secret key to form MAC.

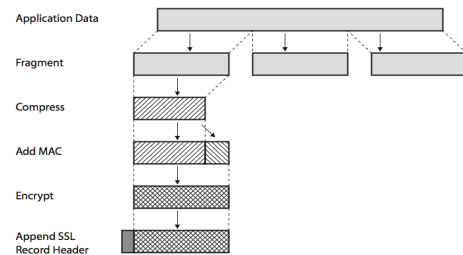
SSL Record Protocol Operation

- Fragmentation:** message is fragmented into blocks of 2^{14} bytes or less
- Compression (optional):** lossless and may not increase the content length by more than 1024 byte (for very short block, it is possible that the output is longer)



SSL Record Protocol Operation

- MAC:** Compute the message authentication code over the compressed data.
- Encryption:** the compressed message plus the MAC are encrypted using symmetric encryption.




Assignment 3

- Due: March 28th (Wed.)
- Done individually or by a group of 2 students


SSL

- Implement a client and a server using **Secure Socket Layer (SSL)**.
- Upon connection, the client prompts the user to enter his/her ID and password.
- After the user enters the ID and the password, the client sends the ID and password to the server through SSL connection.



SSL


- ❖ After the server receives the ID and the password, the server computes the hash of the password, and **prints the ID, the password, and the hashed password.**
- ❖ The server then compares that hashed password against the password stored in file `password`.
 - If the two passwords match, the server sends a string "OK" to the client and the client prints "the password is correct" and terminates
 - otherwise, the client prints "the password is incorrect" and terminates.



SSL

- ❖ The server maintains a file `password` which has the following format:



```
<user ID> <hashed password> <date and time when the password is stored>
```
- ❖ The password can be hashed using `SHA1` or `MD5`. You can use the existing implementation of SHA1 and MD5



SSL

- ❖ You will need to write a program `gen-pass` to generate file `password`.


```
./gen-pass (C/C++)
java Gen-pass (Java)
```
- ❖ When `gen-pass` is invoked, it prompts the person who invokes `gen-pass` to enter each user's ID and password.
- ❖ Your program then saves ID, the hashed password, and the date and time when the password is saved, to file `password`.
- ❖ Your program should also check whether the ID is already in `password`. If so, your program displays "the ID already exists".



SSL


- ❖ The server is invoked as


```
sslserv <server_port> (C/C++)
java SslServ <server_port> (Java)
```

`<server_port>`: the port number on which the server listens for the connection.
- ❖ The client is invoked as



```
sslcli <server_domain> <server_port> (C/C++)
java SslCli <server_domain> <server_port> (Java)
```

`<server_domain>`: the domain name of the server, i.e., `blingsuns.binghamton.edu`




SSL

- ❖ You can use any code available on the web for SSL socket programming and for hashing the password.
- ❖ However, **you must write your own code for the rest part of the assignment** (e.g. enter and verify ID and password, open/read/write files)
- ❖ You should also generate the **certificate** by yourself. Please **use one of your group members' name when generating the certificate** (other information can be forged).




Grading guideline

- ❖ Correct execution format: **2'**
- ❖ Readme: **2'**
- ❖ Makefile: **6'**
- ❖ Correct implementation of `gen-pass/Gen-pass`: **35'**
- ❖ Correct implementation of SSL server and client: **55'**



Submission guideline

- Create a directory with a unique name (e.g. `p3-[userid]`), which contains the source codes, makefile, and a README file.
- **README**
 - ❖ The name and email address of your group members.
 - ❖ Whether your code was tested on bingsuns.
 - ❖ How to execute your program.
 - ❖ (Optional) Briefly describe your algorithm or anything special about your submission that the TA should take note of.



Submission guideline

- Tar the contents of this directory using
`tar -cvf [directory_name].tar [directory_name]`
E.g. `tar -cvf p3-pyang.tar p3-pyang/`
- Upload the tared file you create above to mycourses.