

## SSL/TLS Protocols

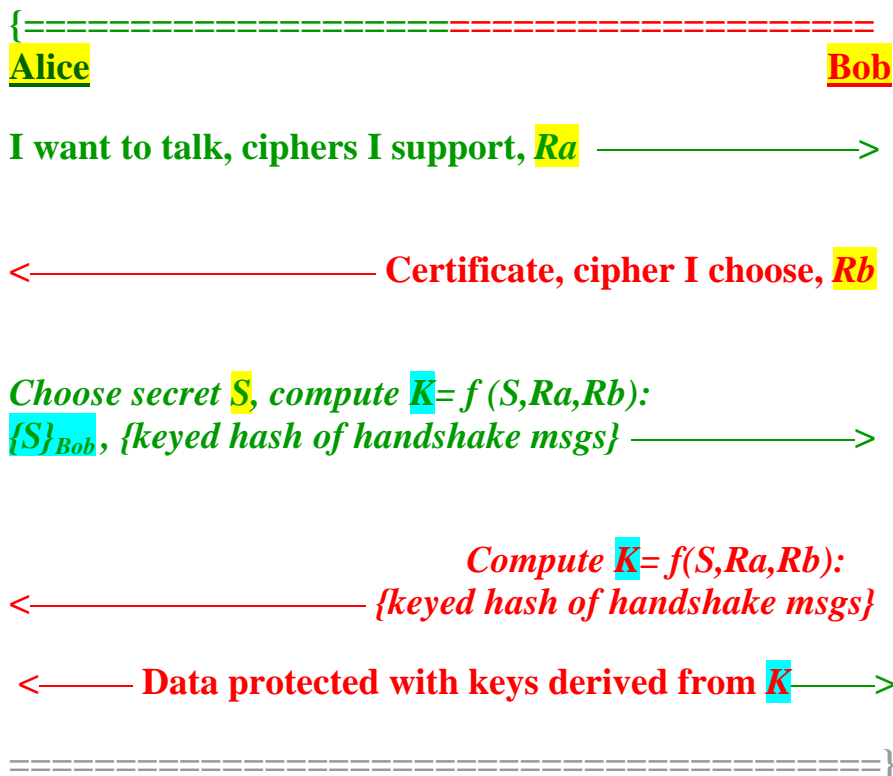
**SSL** (Secure Socket Layer, developed by *Netscape* ) &

**TLS** (Transport Layer Security, is an *IETF* standard)

Are *almost* the same.

They run as a user-level processes on top of *TCP/IP*.

### The Basic Protocol:



**Keys:**

- Alice chooses a random number *S*, as *pre-master secret*.
- It is shuffled with *Ra* and *Rb* to produce a *master secret K*.
- *Ra* and *Rb* are 32 octets long, the first 4 are the UNIX time (number of seconds since Jan 1, 1970).  
This is to ensure that *Rs* are always different.
- The master secret is shuffled with *Rs* to produce six (6) keys:
  - Three (3) for each side for *encryption, integrity, and IV*.
  - The three keys used for transmission are known as the *write keys* while the three keys used for reception are known as the *read keys*
  - Thus Alice's write keys are Bob's read keys and vice versa.
- To ensure that the keyed hash Alice sends is different from the keyed hash Bob sends, Alice include the string "*CLNT*" and the Bob include "*SRVR*" in the hash. (reflection attack!).
- Note that *Alice has authenticated Bob*, but Bob has no idea to whom he's talking to. In SSL it is optional for the server to authenticate the client, if it has a certificate.  
Normally the server authenticates the client using:

*<name, password>*

sent securely over the ssl connection.

## Session Resumption

If the server support session resumption,

it sends *session\_id* for the client.



I want to talk, ciphers I support, *Ra* —————>  
 <————— *session\_id*, certificate, cipher I choose, *Rb*



## Header | encrypted-integrity-protected record

- If block cipher is used, the *IV* is used to encrypt the first record.  
The final block of each record is used as the *IV for the next record*.

### Connection Closure

- The sender should send *close\_notify* message to signal the other end that it has no more data to send.
  - The purpose is to prevent a *truncation attack* in which the attacker inserts a TCP FIN segment before the sender is finished sending data forcing the receiver to think that all data has been received.
  - If a party receives FIN without first receiving *close\_notify* it must mark the session as *not resumable*.
- 

### HTTP Over SSL - https

#### HTTP:

HTTP (HyperText Transfer Protocol) is the Web basic transport protocol.  
The basic unit of HTTP interaction is the *request/response* pair:

- The client opens a TCP connection to the server and writes the request.
- The server writes back the response and indicates the end of response either with a length header or by closing the connection.

#### Example:

- **Client Request:**

GET / HTTP/1.0  
Connection: Keep-Alive  
Host: www.cs.odu.edu

- **Server Response:**

HTTP/1.0 200 OK  
Content-Length: 1650  
Connection: Keep-Alive  
Content-Type: text/html  
.....

**URLs:**

<scheme>://<host>[:<port>]/<path>[?<query>]

**Examples:**

- <scheme>: **http**, default <port> **80**
- <scheme>: **ftp**, default <port> **21**
- <schemes>: **https**, default <port> **443**

**HTTPS:**

The client makes a connection to the server;  
Negotiates an SSL connection; and  
Transmits http data over the established secure connection.

- **Reference integrity:**

Match the URL reference to the server's identity with the **CN name** in the server's PKI certificate.

---

**OpenSSL: s\_server & s\_client**

### s\_server:

```
% openssl s_server -dhparam dh1024.pem -accept 1234 -cert  
server_cert.pem -key server_privatekey.pem
```

```
% openssl s_server -dhparam dh1024.pem -accept 1234 -cert  
server_cert.pem -key server_privatekey.pem -WWW
```

*The option (WWW) causes the server to emulate a simple http server.*

```
% openssl s_server -dhparam dh1024.pem -accept 1234 -cert  
server_cert.pem -key server_privatekey.pem -verify 2 -CAfile  
ca_cert.pem
```

*The option (-verify) causes the server to demand a certificate from the client and the depth of the chain should not exceed 2 and the option (-CAfile) specify the trusted certificate.*

### To create the dh1024.pem:

```
% openssl dhparam -check -text -5 1024 -out dh1024.pem
```

Or use the option `-no_dhe` e.g.:

```
% openssl s_server -no_dhe -accept 1234 -cert server_cert.pem -key  
server_privatekey.pem
```

---

### s\_client:

```
% openssl s_client -connect localhost:1234 -verify 2 -CAfile  
ca_cert.pem
```

```
% openssl s_client -connect localhost:1234 -verify 2 -CAfile  
ca_cert.pem -cert client_cert.pem -key client_privatekey.pem
```

```
% openssl s_client -connect localhost:1234 -verify 2 -CAfile  
ca_cert.pem -reconnect
```

*The option (-reconnect) causes 5 connections to the server using the same session ID to test session caching.*

*To test the WWW mode of server type:*

```
GET /anyfile HTTP/1.0
```