SSL/TLS

How to send your credit card number securely over the internet
The security provided by SSL

- SSL is implemented at level 4
  - The transport control layer
- In practice, SSL uses TCP sockets
  - The underlying TCP implementation handles robustness of communication, such as replay of lost packets, buffering packets to re-order them correctly, etc.
- SSL extends TCP interface for security
What does it entail?

To use SSL, applications must change.

- They have to use the SSL API (application programming interface) and use SSL calls instead of TCP calls.
- Applications’ networking code must change.

SSL may be deployed without making changes to the underlying Operating System, because it does not alter the implementation of the TCP protocol.
The rogue packet problem (1)

- TCP uses checksums to ensure correctness of data.
  - but this checksum prevents only against random errors.
- Suppose an attacker to SSL:
  - Forges the next TCP packet (in a TCP connection, packets are numbered).
  - Re-computes the TCP checksum
    - The TCP protocol accepts the corrupt packet, mark the packet number as delivered/received.
The rogue packet problem (2)

- TCP relays the corrupted packet to SSL
  - SSL checks its cryptographic checksum -- a message authentication code (MAC) -- and realizes that the packet has been forged
- TCP receives true packet from legitimate sender, sees that it has an already used number, and discards the packet as bad.
  - SSL cannot tell TCP to change its behavior, because it has not changed the TCP code. Only option for SSL is to hang up the connection.
- The result is as if the connection had been cut, and the attacker does not need to be “in the middle.”
SSL as software only

Implementing SSL in hardware is unwieldy
- It requires a TCP implementation to function
- Therefore TCP has to be implemented in the same hardware
- But TCP uses long buffers to ensure communication reliability. That means your hardware will require a lot of memory and be costly.

If SSL worked at a lower level, say level 3/ network layer, it could be coded in a net card.
Advantages of SSL

- Allows for portable implementation, because it is an application-level process
  - Suitable for bundling with applications such as browsers, can be installed with user-privileges only, and minimum expertise in anything.
  - SSL can authenticate users (end-to-end authentication), not only machines or IP addresses (link-to-link authentication)
SSL/TLS: First Ingredients

- SSL supports several “cipher suites”:
  - Algorithm sets for public key encryption, symmetric key encryption, and authentication (MACs).
  - Flexibility was needed because of export restrictions.
  - Client and Server must negotiate which algorithms are used in a session.

- Client and server agree on a common secret:
  - Negotiated using public key cryptography
  - Incorporating challenges (nonces) from both parties.
  - From this common secret the symmetric keys are derived.
SSL/TLS: Ingredients (2)

- SSL uses *directional* symmetric keys. After agreeing on common secret:
  - Client and server derive from it two IVs, *encrypt* and *decrypt* keys, as well as *authentication* and *verification* keys. A total of six secrets are derived from the agreed secret (pre-key).
  - Client read keys = Server write keys
    - Server IV = part of Server encryption parameters = part of client decryption parameters
    - Server encryption key = Client decryption key
    - Server authentication key = Client verification key
  - Client write keys = Server read keys
    - Client IV = part of Client encryption parameters = part of server decryption parameters
    - Client encryption key = Server decryption key
    - Client authentication key = Server verification key
SSL/TLS Basic Protocol

Cipher suites supported, $R_1 = R_{Client}$

Choice of cipher suite, $R_2 = R_{Server}$

Client side:
- Create $S$;
- Derive $K$ as $f(S,R_1,R_2)$

Server side:
- Get $S$;
- Derive $K$ as $f(S,R_1,R_2)$

Server:
- $\{S\}_{Server}$, $MAC_{CAK}(handshake)$
- $MAC_{SAK}(handshake)$
- Enc., auth. data