

# **Secure Sockets Layer (SSL) and Transport Layer Security (TLS)**

Halvard Skogsrud

17 October 2002

# Presentation Overview

- Overview of SSL/TLS
- Comparison with IPsec
- Protocol Details
- Issues
- Summary

# Introduction to SSL/TLS

- SSL/TLS allows two parties to establish and mutually authenticate a key (session key) that is used to cryptographically protect the remainder of the session.
- It is a *user-level* process.
- Works on top of the transport layer (OSI model).
- Only works with TCP, *not* UDP.
- Symmetric (shared key) cryptography.

# History

SSL version 1 developed by Netscape, never deployed.

SSL v2 first deployed in Netscape Navigator 1.1 in 1995.

Microsoft improved SSLv2 and proposed a similar protocol, PCT (Private Communications Technology).

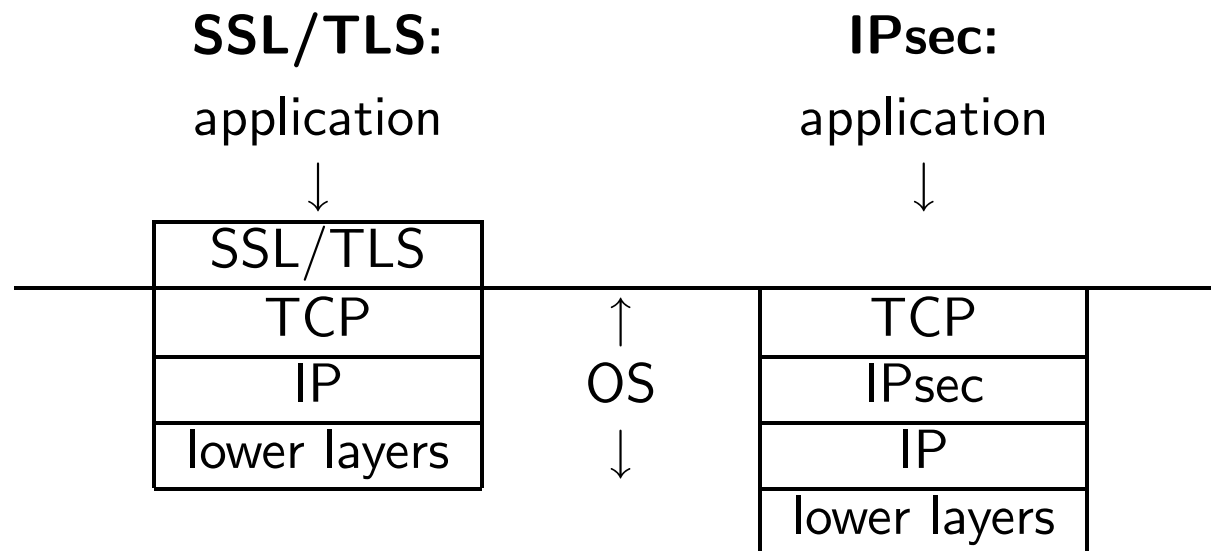
Netscape significantly changed SSL and proposed v3 in Dec 1995.

IETF (Internet Engineering Task Force) introduced a fourth similar but incompatible protocol, TLS, in Apr 1996.

## Comparison SSL/TLS vs. IPsec

What (OSI model) layer is best suited to provide security services?

SSL/TLS provides security on top of the transport layer. IPsec provides much of the same services, but it works on top of the network layer.



## **Advantages of SSL/TLS**

- No need to change OS, only applications change.
- Supports user-level security: Packets to different processes are encrypted differently. Important in time-sharing systems.

## **Disadvantages of SSL/TLS**

- Applications that want to use SSL/TLS must change.

## **Advantages of IPsec**

- Transparency: Applications need not change to support IPsec.

## **Disadvantages of IPsec**

- IPsec must be implemented in the OS.
- Only host-level security: The network layer doesn't know to which processes packets belong, all packets to the same host are encrypted the same way.
- IPsec can support user-level security, but this would require major changes to upper layers, including applications.

## Basic Protocol

1.  $C \rightarrow S$ :  $C$ ,  $\text{Version}_C$ , supported ciphers,  $N_C$
2.  $S \rightarrow C$ :  $\text{Version}_S$ , chosen cipher,  $N_S$ ,  $\text{sign}_{CA}\{S, K_S\}$
3.  $C \rightarrow S$ :  $\{\text{Version}_C, \text{Secret}\}_{K_S}$ ,  $\{\text{hash}(K, \text{handshake messages}, C)\}_K$

Secret is random number,  $K=f(\text{Secret}, N_C, N_S)$

4.  $S \rightarrow C$ :  $\{\text{hash}(K, \text{handshake messages}, S)\}_K$

Further communication protected with keys derived from  $K$ .

In this protocol, the server is authenticated to the client, but the client is not authenticated to the server.

## Client Authentication

1.  $C \rightarrow S$ :  $C$ ,  $\text{Version}_C$ , supported ciphers,  $N_C$
2.  $S \rightarrow C$ :  $\text{Version}_S$ , chosen cipher,  $N_S$ ,  $\text{sign}_{CA}\{S, K_S\}$
3.  $C \rightarrow S$ :  $\text{sign}_{CA}\{C, K_C\}$ ,  $\{\text{Version}_C, \text{Secret}\}_{K_S}$ ,  $\text{sign}_C\{\text{hash}(K, \text{handshake messages}, C)\}$
4.  $S \rightarrow C$ :  $\{\text{hash}(K, \text{handshake messages}, S)\}_K$
5.  **$C \rightarrow S$ :  $\{\text{hash}(K, \text{handshake messages}, C)\}_K$**

Main differences: Client also sends certificate. Client signs hash in message 3 to prove ownership of private key. Extra ClientFinished message at the end, before data transfer starts.

## Resuming Sessions

SSL/TLS: A session is long-lived and can include many connections.

A *session-id* can be included with message 2 if the server wants to allow the session to resume (i.e. use the session for several connections).

Next time the client wants to establish a new connection within the same session, it includes the session-id with message 1. This leads to a shorter handshake:

$C \rightarrow S: Ver_C, \text{session-id}, \text{ciphers}, N_C$

$S \rightarrow C: Ver_S, \text{session-id}, \text{cipher}, N_S, \{\text{hash}(K, \text{messages}, S)\}_K$

$C \rightarrow S: \{\text{hash}(K, \text{handshake messages}, C)\}_K$

## Keys Used

Six different shared keys are computed by various hashes  $\text{hash}_i(K, N_C, N_S)$ .

For each direction, an encryption key, an initialization vector (IV) and a signature key. These keys are recomputed for each connection (with difference nonces).

An intruder never learns the secret (the client encrypts it using the server's public key), and so cannot learn the derived keys.

Nonces are 256 bits long, and it is recommended that the first 32 bits is the Unix time when the message was created.

## Why Is It Secure?

Server certificate means intruder can't pose as server. Client certificates prevents intruder from posing as client.

New keys are computed for each (TCP) connection, so a key compromise will only affect a single connection.

Client and server encrypts and signs all previous handshake messages (i.e. all messages not protected by the negotiated cipher) in messages 3, 4, 5 (ClientVerify, ServerFinished, ClientFinished). This detects any tampering in the handshake messages.

Nonces prevent replay attacks.

## PKI in SSL/TLS

How can the client verify the signature of the public key certificate sent by the server?

Web browsers come with the public keys of various organizations pre-installed. If certificate signature key is not installed in the browser, the user will be asked what to do.

Note that a certificate chain may be submitted instead of a single certificate.

## Version Number Madness

Two octets used for the version number. SSLv2 version field is 0.2, SSLv3 is 3.0 and TLS is 3.1.

SSLv3 moved the position of the version number field!

So an SSLv3 client cannot send a v3 hello message unless it knows the server supports SSLv3, so it sends a v2 hello with version number 3.0.

SSLv2 didn't specify what to do if a higher version number was found. Some implementations just ignore it, and that's what SSLv3 and TLS depend on!

Netscape doesn't learn: If SSLv3 (and TLS) sees a version number higher than what they support, they will ignore it and parse it assuming it's compatible.

## Version Rollback Weakness

Mitchell et al.<sup>1</sup> performed a formal analysis of SSLv3. Procedure: Start with very simple protocol. Use model checking to identify attacks. Add features to eliminate attack. Repeat until no attacks found.

Found weakness in *session resumption*: The intruder changes the version number in the ClientHello to 2.0. In SSLv2, there verification messages don't contain the previous handshake messages. When *resuming* a session, the attack goes unnoticed and the parties use SSLv2.

This makes SSLv3 vulnerable to attacks against SSLv2 (and they exist).

---

<sup>1</sup>J. C. Mitchell, V. Shmatikov, U. Stern. Finite-State Analysis of SSL 3.0. In *7th USENIX Security Symposium*, San Antonio, 1998, pages 201–216

## Other attacks against SSLv3

**Denial-of-Service:** TCP cannot detect a malicious data packet. If an intruder inserts a bogus packet into the data stream, it will be acknowledged by TCP and passed up to SSL. SSL rejects it, but when the real packet comes, TCP will treat it as a duplicate and discard it. SSL has to close the connection in this case.

If SSL was designed to work on top of UDP instead, this attack would not work. However, the design decision was to allow for this attack in order to keep the protocol simple.

## Attacks against SSLv2

SSLv2 is vulnerable to the following attacks:

**Downgrade Attack:** The initial handshake is not integrity-protected, so an attacker can remove strong ciphers from the list of supported ciphers. As a result, the parties would agree on a weak cipher.

**Truncation Attack:** There was no message indicating no more data to be sent, SSLv2 relied on the TCP connection closing. The TCP close message was not cryptographically protected, so an attacker could send a message closing the connection, causing unpredictable behaviour.

## Exportability

While US export restrictions still were in place, 40-bit encryption was the strongest encryption supported by SSL outside US. This was considered insecure, but the fault was not that of SSL, but rather that of the export restrictions.

In the exportable version of SSLv2 the keys were still 128 bits, but only 40 bits were secret. SSLv3 used a 40-bit secret and hashed this together with two nonces to produce a 128 bit key.

## Microsoft SSL Bug

An SSL bug was found in Internet Explorer in Aug 2002<sup>2</sup>. The vulnerability is a bug in Windows, not a bug in the SSL protocol.

If the server submits a credential chain, the client has to verify that the intermediary CAs have certificates stating they are CAs for the domain.

The bug is that IE only checks that the intermediary certificates are signed by the parent CA, it doesn't check that the signer authorized the certificate holder to be an intermediary CA.

Konquerer (KDE web browser) had the same vulnerability, but a fix was available one hour after the flaw was published.

Microsoft has yet to provide a fix for this problem...

---

<sup>2</sup>Source: <http://www.thoughtcrime.org/ie-ssl-chain.txt>

# Summary

- Several protocols exist for transport layer security: SSL (v2 and v3), TLS and PCT.
- No ways to break SSLv3 security discovered, but DoS and version rollback attacks are possible.
- No single standard agreed upon, but SSLv3 most widespread.
- Don't use Internet Explorer for any important transactions involving SSL.