# Transport Layer Security

aus Wikipedia, der freien Enzyklopädie

**Transport Layer Security** (**TLS**; deutsch *Transportschichtsicherheit*), weitläufiger bekannt unter der Vorgängerbezeichnung **Secure Sockets Layer** (**SSL**), ist ein hybrides Verschlüsselungsprotokoll zur sicheren Datenübertragung im Internet. Seit Version 3.0 wird das SSL-Protokoll unter dem neuen Namen TLS weiterentwickelt und standardisiert, wobei Version 1.0 von TLS der Version 3.1 von SSL entspricht. In diesem Artikel wird die Abkürzung TLS für beide Bezeichnungen verwendet, sofern nicht explizit auf die alten Versionen Bezug genommen wird. Bekannte Implementierungen des Protokolls sind OpenSSL und GnuTLS.

TLS im TCP/IP-Protokollstapel

| | | | | | | |
|---|---|---|---|---|---|---|
| *Anwendung* | HTTPS | IMAPS | POP3S | SMTPS | ... | |
| **Transport** | **TLS** | | | | | |
| | **TCP** | | | | | |
| *Internet* | IP | | | | | |
| *Netzzugang* | Ethernet | Token Bus | Token Ring | FDDI | ... | |

## Inhaltsverzeichnis



## TLS in der Praxis

TLS-Verschlüsselung wird heute vor allem mit HTTPS eingesetzt. Die meisten Webserver unterstützen TLS 1.0, viele auch SSLv2 und SSLv3 mit einer Vielzahl von Verschlüsselungsmethoden, fast alle Browser und Server setzen jedoch bevorzugt TLS mit RSA- und AES- oder Camellia-Verschlüsselung ein. In aktuellen Browsern ist SSLv2 deaktiviert oder führt zu einer Sicherheitswarnung,[1] da diese Protokollversion eine Reihe von Sicherheitslücken[2][3] aufweist. Die Weiterentwicklung TLS 1.1 wird von Google Chrome unterstützt, TLS 1.2 wird in der Standardkonfiguration von Apple iOS verwendet und kann im Internet Explorer und Opera aktiviert werden (Stand 08/2012).

Seit einiger Zeit nutzen immer mehr Webseitenbetreiber Extended-Validation-TLS-Zertifikate (EV-TLS-Zertifikat). In der Adresszeile des Browsers wird zusätzlich ein Feld angezeigt, in dem Zertifikats- und Domaininhaber im Wechsel mit der Zertifizierungsstelle (bspw. VeriSign oder TC TrustCenter) eingeblendet werden. Zudem wird je nach verwendetem Browser und/oder Add-on die Adresszeile (teilweise) grün eingefärbt. Internetnutzer sollen so noch schneller erkennen, ob die besuchte Webseite echt ist, und besser vor Phishingversuchen geschützt werden.

TLS wäre ohne die zertifikatsbasierte Authentifizierung anfällig für Man-In-The-Middle-Angriffe: Ist der Man-In-The-Middle vor der Übergabe des Schlüssels aktiv, kann er mit beiden Seiten den Schlüssel tauschen und so den gesamten Datenverkehr im Klartext mitschneiden. Allerdings wird seit Anfang 2010 die Sicherheit von TLS im Zusammenhang von HTTPS mit Hinweis gerade auf die möglicherweise mangelnde Vertrauenswürdigkeit der zertifizierenden Stellen (CAs) angezweifelt.[4][5][6][7]

In Verbindung mit einem *virtuellen Server*, zum Beispiel mit HTTP (etwa beim Apache HTTP Server über den *VHost*-Mechanismus), ist es grundsätzlich als Nachteil zu werten, dass pro Kombination aus IP-Adresse und Port nur ein Zertifikat verwendet werden kann, da die eigentlichen Nutzdaten des darüber liegenden Protokolls (und damit der Name des VHosts) zum Zeitpunkt des TLS-Handshakes noch nicht übertragen wurden. Dieses Problem wurde mit der TLS-Erweiterung Server Name Indication im Juni 2003 durch die RFC 3546 behoben. Dabei wird bereits beim Verbindungsaufbau der gewünschte Servername mitgesendet. Die ursprüngliche Erweiterung wurde für TLS 1.0 beschrieben, aufgrund der Kompatibilität der einzelnen TLS-Versionen zueinander wird SNI auch bei TLS 1.1 und TLS 1.2 entsprechend der Empfehlung umgesetzt.

Weitere bekannte Anwendungsfälle für TLS sind POP3, SMTP, NNTP, SIP, IMAP, XMPP, IRC, LDAP, MBS/IP, FTP, EAP-TLS, TN3270 und OpenVPN.

## Geschichte

- 1994, neun Monate nach der ersten Ausgabe von Mosaic, dem ersten verbreiteten Webbrowser, veröffentlichte Netscape Communications die erste Version von SSL (1.0).
- Fünf Monate später wurde zusammen mit einer neuen Ausgabe des Netscape Navigator die nächste Version SSL 2.0 veröffentlicht.
- Ende 1995 kam Microsoft mit der ersten Version seines Browsers (Internet Explorer) heraus. Kurz darauf wurde auch die erste Version ihres SSL-Pendants bekannt, PCT 1.0 (Private Communication Technology). PCT hatte einige Vorteile gegenüber SSL 2.0, die später in SSL 3.0 aufgenommen wurden.
- Als SSL von der IETF im RFC 2246 als Standard festgelegt wurde, benannte man es im Januar 1999 um zu *Transport Layer Security (TLS)*. Die Unterschiede zwischen SSL 3.0 und TLS 1.0 sind nicht groß. Doch dadurch entstanden Versionsverwirrungen. So meldet sich TLS 1.0 im Header als Version SSL 3.1.
- Später wurde TLS durch weitere RFCs erweitert:
  - RFC 2712 – Addition of Kerberos Cipher Suites to Transport Layer Security (TLS).
  - RFC 2817 – Upgrading to TLS Within HTTP/1.1 erläutert die Benutzung des Upgrade-Mechanismus in HTTP/1.1, um Transport Layer Security (TLS) über eine bestehende TCP-Verbindung zu initialisieren. Dies erlaubt es, für unsicheren und für sicheren HTTP-Verkehr die gleichen „well-known“ TCP Ports (80 bzw. 443) zu benutzen.
  - RFC 2818 – HTTP Over TLS trennt sicheren von unsicherem Verkehr durch Benutzung eines eigenen Server-TCP-Ports.
  - RFC 3268 – Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS) nutzt die Erweiterbarkeit von TLS und fügt den bisher unterstützten symmetrischen Verschlüsselungsalgorithmen (RC2, RC4, International Data Encryption Algorithm (IDEA), Data Encryption Standard (DES) und Triple DES) den Advanced Encryption Standard (AES) hinzu.
  - RFC 3546 - Transport Layer Security (TLS) Extensions führt das Konzept der Erweiterungen ein durch welches optionale Datenfelder/Header vor allem bei der

anfänglichen Aushandlung übertragen werden können, eine der wohl bekanntesten Erweiterung ist Server Name Indication
- Im April 2006 wurde in RFC 4346 die Version 1.1 von TLS standardisiert und damit RFC 2246 obsolet. In TLS 1.1 wurden kleinere Sicherheitsverbesserungen vorgenommen und Unklarheiten beseitigt.
- Im August 2008 erschien mit RFC 5246 die Version 1.2 von TLS, welche somit RFC 4346 obsolet machte. Als wesentliche Änderung wurde die Festlegung auf MD5/SHA-1 in der Pseudozufallsfunktion (PRF) und bei signierten Elementen fallen gelassen. Stattdessen wurden flexiblere Lösungen gewählt, bei denen die Hash-Algorithmen spezifiziert werden können.

## Funktionsweise

Der Client baut eine Verbindung zum Server auf. Für gewöhnlich authentifiziert sich zuerst der Server gegenüber dem Client mit einem Zertifikat. Dann schickt entweder der Client dem Server eine mit dem öffentlichen Schlüssel des Servers verschlüsselte geheime Zufallszahl, oder die beiden Parteien berechnen mit dem Diffie-Hellman-Schlüsselaustausch ein gemeinsames Geheimnis. Aus dem Geheimnis wird dann ein kryptographischer Schlüssel abgeleitet. Dieser Schlüssel wird in der Folge benutzt, um alle Nachrichten der Verbindung mit einem symmetrischen Verschlüsselungsverfahren zu verschlüsseln und zum Schutz von Nachrichten-Integrität und Authentizität durch einen Message Authentication Code abzusichern.

### TLS-Protokolle im Protokollstapel

Im OSI-Modell ist TLS in Schicht 5 (der Sitzungsschicht) angeordnet. Im TCP/IP Modell ist TLS oberhalb der Transportschicht (zum Beispiel TCP) und unterhalb Anwendungsprotokollen wie HTTP oder SMTP angesiedelt. In den Spezifikationen wird dies dann zum Beispiel als „*HTTP over TLS*“ bezeichnet. Sollen jedoch beide Protokolle zusammengefasst betrachtet werden, wird üblicherweise ein „*S*“ *für Secure* dem Protokoll der Anwendungsschicht angehängt (zum Beispiel HTTP**S**). TLS arbeitet transparent, so dass es leicht eingesetzt werden kann, um Protokollen ohne eigene Sicherheitsmechanismen abgesicherte Verbindungen zur Verfügung zu stellen. Zudem ist es erweiterbar, um Flexibilität und Zukunftssicherheit bei den verwendeten Verschlüsselungstechniken zu gewährleisten.

Das TLS-Protokoll besteht aus zwei Schichten:

| **TLS Handshake Protocol** | **TLS Change Cipher Spec. Protocol** | **TLS Alert Protocol** | **TLS Application Data Protocol** |
|---|---|---|---|
| **TLS Record Protocol** | | | |

### TLS Record Protocol

Das TLS Record Protocol ist die untere der beiden Schichten und dient zur Absicherung der Verbindung. Es setzt direkt auf der Transportschicht auf und bietet zwei verschiedene Dienste, die einzeln oder gemeinsam genutzt werden können:

- Ende-zu-Ende-Verschlüsselung mittels symmetrischer Algorithmen. Der verwendete Schlüssel wird dabei im Voraus über ein weiteres Protokoll (zum Beispiel das TLS Handshake Protocol) ausgehandelt und kann nur einmal für die jeweilige Verbindung verwendet werden. TLS unterstützt für die symmetrische Verschlüsselung unter anderem DES, Triple DES und AES
- Sicherung der Nachrichten-Integrität und Authentizität durch einen Message Authentication Code, in der Regel HMAC.

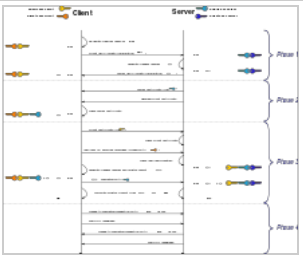
Außerdem werden zu sichernde Daten in Blöcke von maximal 16.384 ($2^{14}$) Byte fragmentiert und beim Empfänger wieder zusammengesetzt. Dabei schreibt der Standard vor, dass die Blockgröße diesen Wert nicht übersteigt, außer der Block ist komprimiert oder verschlüsselt - dann darf die Blockgröße um 1024 Byte (bei Kompression) bzw. 2048 Byte (bei Verschlüsselung) größer sein. Auch können die Daten vor dem Verschlüsseln und vor dem Berechnen der kryptografischen Prüfsumme komprimiert werden. Das Komprimierungsverfahren wird ebenso wie die kryptografischen Schlüssel mit dem TLS Handshake-Protokoll ausgehandelt.

Der Aufbau einer TLS Record Nachricht lautet wie folgt: **Content Type** (1 Byte: *Change Cipher Spec* = 20, *Alert* = 21, *Handshake* = 22, *Application Data* = 23) | **Protokollversion Major** (1 Byte) | **Protokollversion Minor** (1 Byte) | **Länge** (1 Short bzw. zwei Byte)

### TLS Handshake Protocol

Das TLS Handshake Protocol baut auf dem TLS Record Protocol auf und erfüllt die folgenden Funktionen, noch bevor die ersten Bits des Anwendungsdatenstromes ausgetauscht wurden:

- Identifikation und Authentifizierung der Kommunikationspartner auf Basis asymmetrischer Verschlüsselungsverfahren und Public-Key-Kryptografie. Dieser Schritt ist optional eine Zwei-Wege-Authentifizierung, für gewöhnlich authentifiziert sich aber nur der Server gegenüber dem Client.
- Aushandeln zu benutzender kryptografischer Algorithmen und Schlüssel. TLS unterstützt auch eine unverschlüsselte Übertragung.



TLS-Handshake mit Zwei-Wege-Authentifizierung mittels Zertifikaten

Der Handshake selbst kann in vier **Phasen** unterteilt werden:

1. Der Client schickt zum Server ein *client_hello*, und der Server antwortet dem Client mit einem *server_hello*. Die Parameter der Nachrichten sind:
   - die Version (die höchste vom Client unterstützte TLS-Protokoll-Version)
   - eine 32 Byte Zufallsinformation (4 Byte Timestamp + 28 Byte lange Zufallszahl), die später verwendet wird, um das *pre-master-secret* zu bilden (sie schützt damit vor Replay-Attacken)
   - eine Session-ID
   - die zu verwendende Cipher Suite (Algorithmen für Schlüsselaustausch, Verschlüsselung und Authentifizierung)
   - Optional den gewünschten FQDN für die Unterstützung von Server Name Indication
2. Diese Phase darf nur bei anonymem *Key Agreement* weggelassen werden. Der Server identifiziert sich gegenüber dem Client. Hier wird auch das X.509v3-Zertifikat zum Client übermittelt. Außerdem kann der Server einen *CertificateRequest* an den Client schicken.
3. Hier identifiziert sich der Client gegenüber dem Server. Besitzt der Client kein Zertifikat, antwortete er früher mit einem *NoCertificateAlert*. TLS-konforme Systeme verwenden diesen Alert jedoch nicht. Der Client versucht außerdem, das Zertifikat, das er vom Server erhalten hat, zu verifizieren (bei Misserfolg wird die Verbindung abgebrochen). Dieses Zertifikat enthält den öffentlichen Schlüssel des Servers. Wird die Cipher-Suite RSA verwendet, so wird das vom Client generierte *pre-master-secret* mit diesem öffentlichen Schlüssel verschlüsselt und kann vom Server mit dem nur ihm bekannten privaten Schlüssel wieder entschlüsselt werden. Alternativ kann hier auch das Diffie-Hellman-Verfahren verwendet werden, um ein gemeinsames *pre-master-secret*, das die Voraussetzungen für Folgenlosigkeit erfüllt, zu generieren. Diese Phase ist optional.
4. Diese Phase schließt den Handshake ab. Aus dem vorhandenen *pre-master-secret* kann das *Master Secret* abgeleitet werden und aus diesem der einmalige Sitzungsschlüssel (englisch „session key“). Das ist ein einmalig benutzbarer symmetrischer Schlüssel, der während der Verbindung zum Ver- und Entschlüsseln der Daten genutzt wird. Die Nachrichten, die die Kommunikationspartner sich nun gegenseitig zusenden, werden nur noch verschlüsselt übertragen.

### TLS Change Cipher Spec Protocol

Das Change Cipher Spec Protocol besteht nur aus einer einzigen Nachricht. Diese Nachricht ist 1 Byte groß und besitzt den Inhalt 1. Durch diese Nachricht teilt der Sender dem Empfänger mit, dass er in der aktiven Sitzung auf die im Handshake Protocol ausgehandelte Cipher Suite wechselt.

### TLS Alert Protocol

Das Alert Protocol unterscheidet etwa zwei Dutzend verschiedene Mitteilungen. Eine davon teilt das Ende der Sitzung mit (close_notify). Andere beziehen sich zum Beispiel auf die Protokollsyntax oder die Gültigkeit der verwendeten Zertifikate. Es wird zwischen Warnungen und Fehlern unterschieden, wobei Letztere die Verbindung sofort beenden.

Der Aufbau einer Fehlermeldung lautet wie folgt: **AlertLevel** (1 Byte: *Warning* = 1, *Fatal* = 2) | **AlertDescription** (1 Byte: *close_notify* = 0, [...], *no_renegotiation* = 100).

In der Spezifikation von TLS werden die folgenden schweren Fehlertypen definiert:

| | |
|---|---|
| unexpected_message | Unpassende Nachricht wurde empfangen |
| bad_record_mac | Ein falscher MAC wurde empfangen |
| decompression_failure | Dekomprimierungsalgorithmus empfing unkorrekte Daten |
| handshake_failure | Absender konnte keine akzeptable Menge von Sicherheitsparametern bearbeiten. |
| illegal_parameter | Ein Feld in der Handshake-Nachricht lag außerhalb des erlaubten Bereichs oder stand im Widerspruch mit anderen Feldern |

In der Spezifikation von TLS werden die folgenden Warnungen definiert:

| | |
|---|---|
| close_notify | Teilt Empfänger mit, dass Absender keine weiteren Nachrichten auf dieser Verbindung senden wird. Muss von jedem Partner einer Verbindung als letzte Nachricht gesendet werden. |
| no_certificate | Kann als Antwort auf eine Zertifikatanforderung gesendet werden, falls passendes Zertifikat nicht verfügbar ist. (Wurde in TLS 1.0 entfernt[8]) |
| bad_certificate | Empfangenes Zertifikat war unvollständig oder falsch. |
| unsupported_certificate | Der Typ des empfangenden Zertifikats wird nicht unterstützt. |
| certificate_revoked | Zertifikat wurde vom Unterzeichner zurückgerufen. |
| certificate_expired | Zertifikat ist abgelaufen. |
| certificate_unknown | Andere nicht genau spezifizierte Gründe sind beim Bearbeiten des Zertifikats aufgetreten, die dazu führen, dass das Zertifikat als ungültig gekennzeichnet wurde. |

In der Spezifikation von TLS 1.0 wurden folgende Warnungen ergänzt[8]:

| | |
|---|---|
| decryption_failed | Entschlüsselung fehlgeschlagen |
| record_overflow | |
| unknown_ca | Unbekannte oder nicht vertrauenswürdige CA. |
| access_denied | Zugriff verweigert. |
| decode_error | Decodierungsfehler. |
| decrypt_error | Entschlüsselungsfehler. |
| export_restriction | |
| protocol_version | Veraltete Version von TLS/SSL |
| insufficient_security | Unausreichende Sicherheit |
| internal_error | Interner Fehler. |
| user_canceled | Abbruch durch Benutzer |
| no_renegotiation | |

### TLS Application Data Protocol

Die Anwendungsdaten werden über das Record Protocol transportiert, in Teile zerlegt, komprimiert und in Abhängigkeit vom aktuellen Zustand der Sitzung auch verschlüsselt. Inhaltlich werden sie von TLS nicht näher interpretiert.

### Berechnung des Master Secrets

Aus dem pre-master-secret wird in früheren Protokollversionen mit Hilfe der Hash-Funktionen SHA-1 und MD5, in TLS 1.2 mit Hilfe einer durch eine Cipher Suite spezifizierten Pseudozufallsfunktion das Master Secret berechnet. In diese Berechnung fließen zusätzlich die Zufallszahlen der Phase 1 des Handshakes mit ein. Die Verwendung beider Hash-Funktionen sollte sicherstellen, dass das Master Secret immer noch geschützt ist, falls eine der Funktionen als kompromittiert gilt. In TLS 1.2 wird dieser Ansatz nun durch die flexible Austauschbarkeit der Funktion ersetzt.

## Vor- und Nachteile

Der Vorteil des TLS-Protokolls ist die Möglichkeit, jedes höhere Protokoll auf Basis des TLS-Protokolls zu implementieren. Damit ist eine Unabhängigkeit von Anwendungen und Systemen gewährleistet.

Der Nachteil der TLS-verschlüsselten Übertragung besteht darin, dass der Verbindungsaufbau auf Serverseite rechenintensiv und deshalb langsamer ist. Die Verschlüsselung selbst nimmt je nach verwendetem Algorithmus nur noch wenig Rechenzeit in Anspruch. Die verschlüsselten Daten können von transparenten Kompressionsverfahren (etwa auf PPTP-Ebene) kaum mehr komprimiert werden. Als Alternative bietet das TLS-Protokoll ab Version 1.0 die Option, die übertragenen Daten mit zlib zu komprimieren, dies wird jedoch zurzeit (08/2012) nur von Google Chrome unterstützt.

TLS verschlüsselt nur die Kommunikation zwischen zwei Stationen. Es sind jedoch auch Szenarien (insbesondere in serviceorientierten Architekturen) denkbar, in denen eine Nachricht über mehrere Stationen gesendet wird. Wenn jede dieser Stationen aber nur einen Teil der Nachricht lesen darf, reicht TLS nicht mehr aus, da jede Station alle Daten der Nachricht entschlüsseln kann. Somit entstehen Sicherheitslücken an jeder Station, die nicht für sie bestimmte Daten entschlüsseln kann.

## Siehe auch

- Kryptografie
- Netzwerkprotokoll
- Elektronische Unterschrift
- Datagram Transport Layer Security
- IPsec
- Comparison of TLS implementations

## Literatur


- Eric Rescorla: *SSL and TLS. Designing and building secure systems.* Addison-Wesley, New York NY u. a. 2001, ISBN 0-201-61598-3.
- Roland Bless u. a.: *Sichere Netzwerkkommunikation. Grundlagen, Protokolle und Architekturen.* Springer Verlag, Berlin u. a. 2005, ISBN 3-540-21845-9 (*X.systems.press*).
- Claudia Eckert: *IT-Sicherheit. Konzepte – Verfahren – Protokolle.* 6. überarbeitete Auflage. Oldenbourg, München u. a. 2009, ISBN 978-3-486-58999-3.


## Weblinks

- TLS-Arbeitsgruppe (http://www.ietf.org/html.charters/tls-charter.html) der IETF
- TLS 1.2 Spezifikation (Proposed Standard) (http://tools.ietf.org/rfcmarkup/5246) der IETF (TLS-Arbeitsgruppe)
- SSL 3.0 Spezifikation (http://web.archive.org/web/20080208141212/http://wp.netscape.com/eng/ssl3/)
- Einführung in SSL von Markus Repges (http://www.repges.net/SSL/ssl.html) Beschreibt Handshake und Protokoll im Detail

## Einzelnachweise


1. http://support.mozilla.com/de/kb/Firefox%20kann%20keine%20sichere%20Verbindung%20aufbauen%20weil%20die%20Website%20eine%20%C3%A4ltere%20unsiche Protokolls%20verwendet
2. http://lists.alioth.debian.org/pipermail/pkg-mozilla-maintainers/2005-April/000024.html
3. http://www.gnutls.org/manual/html_node/On-SSL-2-and-older-protocols.html
4. http://www.heise.de/newsticker/meldung/EFF-zweifelt-an-Abhoersicherheit-von-SSL-963857.html
5. http://www.eff.org/deeplinks/2010/03/researchers-reveal-likelihood-governments-fake-ssl
6. http://files.cloudprivacy.net/ssl-mitm.pdf
7. http://www.wired.com/threatlevel/2010/03/packet-forensics/
8. Schwenk, Jörg (2010): *Sicherheit und Kryptographie im Internet. Von sicherer E-Mail bis zu IP-Verschlüsselung*, herausgegeben von Vieweg+Teubner Verlag / GWV Fachverlage GmbH, Wiesbaden. ISBN 978-3-8348-0814-1