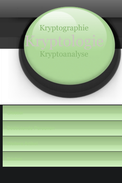
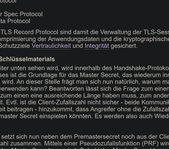


Transport Layer Security (TLS)

Die **Transport Layer Security** (Transportsicherheit, kurz **TLS**) ist der **Nachfolger des Secure Sockets Layer** (SSL). Konkret ist es ein hybrides Verschlüsselungsprotokoll für eine sichere Kommunikation im Internet. Am Häufigsten wird man mit TLS wohl über HTTPS in Kontakt kommen, wo das Verschlüsselungsverfahren eingesetzt wird.

TLS-Protokollstruktur



Funktionsweise und Ablauf

Der generelle Ablauf bei TLS beginnt mit dem Aufbau einer Verbindung vom Client zum Server. Dabei schickt er gleich eine Liste an unterstützten Cipher Suites mit. Anschließend authentisiert sich der Server gegenüber dem Client mit einem Zertifikat und der ausgewählten Cipher Suite. Der Client überprüft das Zertifikat und authentisiert sich ggf. selbst auch noch gegenüber dem Server mit einem eigenen Zertifikat. Nun schickt entweder der Client dem Server eine verschlüsselte Zufallszahl, die mit dem öffentlichen Schlüssel des Servers verschlüsselt ist, oder beide Parteien berechnen ein gemeinsames Geheimnis mit dem Diffie-Hellman-Schlüsselaustauschverfahren. Mit dem daraus abgeleiteten kryptographischen Schlüssel, werden nun alle Nachrichten der Verbindung mit einem ausgewählten symmetrischen Verschlüsselungsverfahren verschlüsselt. Folgende Komponenten sind bei TLS dabei aktiv:

TLS Record Protocol

Das TLS Record Protocol ist ein Schichtenprotokoll, dass die zu sendenden Nachrichten nimmt, die Daten in Blöcke unterteilt (fragmentiert), diese optional komprimiert (Default nicht, ansonsten wird Komprimierungsverfahren im Handshake ausgehandelt), MAC darauf anwendet (für den Integritätsschutz), verschlüsselt (symmetrisch Verschlüsselung – Stromchiffren, Blockchiffren oder kombinierte Verfahren) und das Ergebnis verschickt. Empfangene Daten werden entschlüsselt, verifiziert, dekomprimiert und wieder zusammengesetzt. Vier Protokolle bauen auf dem TLS Record Protocol auf:

- Handshake Protocol
- Alert Protocol
- Change Cipher Spec Protocol
- Application Data Protocol

Kernaufgaben des TLS Record Protocol sind damit die Verwaltung der TLS-Session, die Fragmentierung/Komprimierung der Anwendungsdaten und die kryptographische Verarbeitung. Damit werden die Schutzziele Vertraulichkeit und Integrität gesichert.

Berechnung des Schlüsselmaterials

Wie man etwas weiter unten sehen wird, wird innerhalb des Handshake-Protokolls das Premaster Secret erstellt. Dieses ist die Grundlage für das Master Secret, das wiederum innerhalb des Record Protocols generiert wird. An dieser Stelle fragt man sich nun natürlich, warum man nicht gleich das Premaster Secret verwenden kann? Beantworten lässt sich die Frage zum einen dadurch, dass das Master Secret zum einen eine ausreichende Länge haben muss, zum anderen erhält man damit die Sicherheit. Evtl. ist ein Client-Zufallszahl nicht sicher - beide Kommunikationsparteien wollen zur Sicherheit beitragen - hinzukommt, dass Angreifer ohne die Zufallszahlen das verschlüsselte Premaster Secret einspielen könnten. Es werden also auch Wiederholungsangriffe vorgebeugt.

Das Master Secret setzt sich nun neben dem Premastersecret noch aus der Client Zufallszahl und der Server Zufallszahl zusammen. Mittels einer Pseudozufallsfunktion (PRF) wird aus diesen drei Bestandteilen nun das Master Secret generiert. Sobald das Master Secret berechnet wurde, sollte der Pre Master Secret aus dem Speicher gelöscht werden.

Berechnung des Master Secrets:

```
master_secret = PRF(pre_master_secret, "master secret", ClientHello.random + ServerHello.random)
```

Das Master Secret ist immer genau 48 Byte lang.

TLS Handshake Protocol

Das TLS Handshake Protocol ist für die Aushandlung einer Session (und damit für die dafür verwendeten Sicherheits-Parameter) zuständig. Die Handshake-Nachrichten werden dann dem TLS Record Layer geliefert, wo sie zusammen mit den anderen Daten entsprechend verarbeitet werden. Das TLS Handshake Protocol besteht folgende Aufgaben:

- Aushandlung der verwendeten kryptographischen Algorithmen (für Verschlüsselung & Integrität) – aber auch unverschlüsselte Übertragung möglich
- Authentifizierung der Kommunikationspartner (Server und/oder Client – meistens aber nur Server gegenüber Client)
- Aushandlung des Schlüsselmaterials

Ablauf (vereinfacht)

ClientHello

Das Handshake-Protokoll beginnt mit einem ClientHello vom Client an den Server. Darin ist eine Liste an Cipher Suites enthalten, eine 32 Byte lange Nonce des Clients (28 Byte Zufallszahl + 7 Byte aktuelle Zeit des Client) sowie eine Sitzungsnummer für eine evtl. spätere Sitzungswiederaufnahme. Zusätzlich wir eine Liste an Kompromierungsalgorithmen gesendet, die der Client unterstützt.

ServerHello

Auf einen ClientHello muss der Server mit einem ServerHello antworten. Geschieht dies nicht, kommt es zu einem Fatal Error und die Verbindung schlägt fehl. Die ServerHello-Antwort enthält neben der ausgewählten CipherSuite, die Nonce des Servers und eine gewählte Sitzungsnummer.

Server Certificate

Diese Nachricht wird benötigt, falls die auswählte Schlüsselaustausch-Methode ein Zertifikat vorsieht. Die Server Certificate Nachricht folgt immer direkt der ServerHello-Nachricht. Gesendet wird letztendlich das x509 Zertifikat, passend zur gewählten CipherSuite.

ServerKeyExchange

Die ServerKeyExchange-Nachricht wird nur gesendet, falls das Server Zertifikat nicht alle Parameter zur Schlüsselberechnung (des Premaster Secrets) enthält. Dies ist der Fall bei:

- DHE_DSS
- DHE_RSA
- DH_anon

CertificateRequest

Auch die Nachricht Certificate-Request ist optional. Mit dieser Nachricht kann auch der Server vom Client ein Zertifikat fordern, falls dieses für die ausgewählte Cipher Suite zwecknmäßig ist. Gesendet wird die Certificate Request Nachricht gleich nach der ServerKeyExchange-Nachricht, bzw. falls diese nicht gesendet wird, nach der Nachricht mit dem Zertifikat des Servers.

Server Hello Done

Die ServerHelloDone-Nachricht ist der Abschluss der Authentifizierung und des Schlüsselaustausch des Servers. Sie zeigt das Ende der mit dem ServerHello einhergehenden Kommunikation an. Nun wartet der Server auf die Antwort des Clients.

Bevor der Client aber nun seinerseits mit Nachrichten beginnt, sollte er erst einmal verifizieren, ob das Zertifikat des Servers valide ist. Auch die anderen Parameter der Server Hello-Nachrichten sollten überprüft werden.

ClientCertificate

Falls es eine Aufforderung des Servers gab, sendet mit der ClientCertificate-Nachricht der Client seinerseits sein X509 Zertifikat, passend zur gewählten CipherSuite. Falls kein passendes Zertifikat vorhanden ist, muss der Client dennoch eine Certificate-Nachricht senden. Sendet er diese Nachricht nicht, kann der Server die Verbindung abbrechen oder aber auch den Handshake ohne Client-Authentifizierung weiter ausführen.

Client Key Exchange

Während die ServerKey-Exchange-Nachricht optional ist, wird die ClientKeyExchange-Nachricht immer versendet. Entweder folgt sie sofort der Zertifikats-Nachricht, oder sie ist die erste Nachricht vom Client, die auf der ServerHelloDone-Nachricht folgt. Mit dieser Nachricht wird das Premaster Secret festgelegt, entweder direkt durch das Senden eines RSA-verschlüsselten Secrets oder durch die Übermittlung der Diffie-Hellman-Parameter.

Certificate Verify

Optional ist hingegen wieder die CertificateVerify-Nachricht. Damit findet eine explizite Verifikation des Client-Zertifikats statt, d.h. der Nachweis, dass der Client auch im Besitz des privaten Schlüssels ist und damit der ist, für den er sich ausgibt. Diese Nachricht wird nur versandt, falls das Zertifikat des Clients überhaupt signierbar ist.

Change Cipher Spec

Mit der Nachricht ChangeCipherSpec wird angezeigt, das ab nun gesichert kommuniziert wird. Ab diesem Zeitpunkt nutzt der TLS Record Layer die ausgehandelten kryptographischen Algorithmen und Schlüssel. Die ChangeCipherSpec-Nachricht ist damit gleichzeitig auch die erste geschützte Nachricht.

Finished

Mit der Finished-Nachricht wird bestätigt, das der Schlüsselaustausch und der Authentifizierungsprozess erfolgreich waren. Der Empfänger der Finished-Nachricht muss den Inhalt auf Korrektheit der Nachricht verifizieren. Dafür wird jeweils ein Hashwert über alle ausgetauschte Nachrichten berechnet und überprüft. Damit wird unter anderem eine Downgrade-Attacke vorgebeugt, falls beispielsweise ein Angreifer starke Cipher-Suites aus dem ClientHello gelöscht hatte.

TLS-Alert Protokoll

Wie der Name schon sagt, ist das Alert Protocol für Alerts zuständig. Diese werde in zwei Stufen unterteilt: „Warning" und „Fatal". Fatale Alert Messages führen zu einem sofortigen Abbruch der Verbindung. Die Alert-Nachrichten sind, wie die anderen Nachrichten, verschlüsselt und komprimiert.

Das TLS-Alert Protokoll kennt verschiedene Alerts. Einer der wichtigsten Alerts ist die CloseNotify-Nachricht, da bei TLS ein explizter Verbindungsabbau notwendig ist. Sonst könnte es beispielsweise zu Truncation Angriffe kommen. Jede Partei kann den Austausch einer Closing-Nachricht initiieren und damit der Gegenstelle signalisieren, dass der Sender dem Empfänger keine Nachricht mehr über diese Verbindung schicken wird. Alle Daten, die anschließend empfangen werden, werden ignoriert. Der Empfänger muss seinerseits mit einem close_notify Alert antworten, der Initiator muss allerdings nicht auf diese Antwort warten.

Datagram Transport Layer Security (DTLS)

TLS findet man oberhalb eines zuverlässigen Transportprotokolls vor (z.B. im TCP/IP Modell oberhalb der Transportschicht, z.B über TCP). Ein zuverlässiges Transportprotokoll, wie eben TCP wird vor allem für das Handshake Protocol und dem Record Protocol benötigt. Die Zuverlässigkeit des Handshakes muss garantiert sein, da Paketverluste sonst immer wieder zu einem neuen Handshake führen. Beim Record Protocol hingegen spielt die kryptographische Verarbeitung der Pakete eine Rolle, die nicht unabhängig voneinander möglich ist.

Damit die Anwendung von TLS aber auch mit unzuverlässigen Transportprotokollen, wie beispielsweise UDP, möglich ist, wurde das **Datagram Transport Layer Security (DTLS)** erfunden. Nötig wurde dies, weil immer mehr Anwendungen vermehrt auf UDP zurückgreifen, beispielsweise Voice over IP (VoIP).

DTLS unterscheidet sich dabei nicht grundlegend von TLS, sondern wurde lediglich an den Stellen angepasst, an denen Änderungen bzw. der Verwendung eines nicht zuverlässigen Transportprotokolls notwendig machten. So wurde das Handshake Protocol modifiziert, Sequenznummern im Record Protocol eingeführt, Stromchiffren im Record Layer gestrichen und der Umgang mit verlorenen Paketen überarbeitet.

Cipher-Suites

TLS ist nicht TLS. Denn hinter TLS stehen sogenannte Cipher-Suites und sie bestimmen am Ende, ob die eingesetzte Transportverschlüsselung am Ende auch sicher ist oder nicht. Eine Cipher-Suite ist eine Sammlung kryptografischer Verfahren zur Absicherung von TLS. In einer perfekten Welt würde es nur eine Cipher-Suite geben: Die sicherste mit den besten Verfahren und längsten Schlüssel! Leider leben wir nicht in einer perfekten Welt, denn nicht alle Clients verstehen auch immer sofort die neusten Verfahren oder auch auf Seiten des Servers kann Nachholbedarf angesagt sein. Deshalb unterstützt der Server der TLS in der Regel mehrere Verfahren. Der Client teilt dem Server mit, welche Verfahren er unterstützt und der Server wählt davon dann ein Verfahren aus (siehe für die einzelnen Schritte auch die vorherige Ablauf-Beschreibung von TLS).

Problematisch wird es nun, wenn man sichere aber auch unsichere Verfahren anbietet. Dann kann ein böser Angreifer mit einer sogenannten Downgrade-Attacke ein unsicheres Verfahren vom Server erzwingen. Hierfür kann ein Man-in-the-Middle-Angreifer beispielsweise einfach die unterstützte Verfahren des Clients manipulieren. Bei der Wahl der richtigen Cipher-Suites geht es also auf der einen Seite darum möglichst nur sichere Verfahren anzubieten und auf der anderen Seite aber auch möglichst niemand dadurch auszusperren.

> **Hinweis:** Empfohlen werden die TLS-Versionen 1.2 und 1.3. Alle Vorgänger, wie TLS 1.1 und SSL 2 und SSL 3 sollten auf keinen Fall eingesetzt und abgeschaltet werden!

Die Wahl der richtigen Cipher-Suite ist vor allem für TLS 1.2 relevant. Eine Cipher-Suite ist dabei nach folgendem Schema aufgebaut:

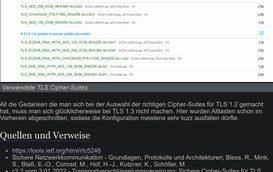
```
TLS_[Schlüsselaustauschverfahren]_WITH_[Verschlüsselungsverfahren]_[Nachrichtenauthentifizierung]
```

Die Suite besteht damit aus den drei Komponenten:

1. **Schlüsselaustauschverfahren**. Dient der Authentifizierung dem Server gegenüber dem Client und wird für die Einigung des gemeinsamen Schlüssels benötigt. Das bekannteste Schlüsselaustauschverfahren ist der Diffie-Hellmann-Schlüsselaustausch.
2. **Verschlüsselungsverfahren**. Die symmetrische Verschlüsselung ist der Kern von TLS und in der Regel wird hier auf AES (Advanced Encryption Standard) in verschiedenen Schlüssellängen (z.B. 128 oder 256 Bit) zurückgegriffen.
3. **Hashverfahren**, das im Message-Authentication-Code (MAC) genutzt wird. Damit wird die Integrität der übertragenen Daten sichergestellt. Die Hashverfahren MD5 oder SHA (ohne Nummer) sollten auf keinen Fall genutzt werden. Aktuell empfiehlt sich der Einsatz von SHA-256 und SHA-384.

> **Hinweis:** Falls man bei der Konfiguration der Cipher-Suite das Wort „ANON" finden, dann ist Handlungsbedarf angesagt. ANON steht für „keine Authentifizierung" und ist ein Sicherheits-GAU. Diese Ciphers müssen ersatzlos entfernt werden. Statt ANON sollte ein Verfahren stehen, dass den Server authentifiziert, z.B. DH_RSA (eine Kombination aus Diffie-Hellmann und RSA). In der Technische Richtlinie TR-02102-2 Kryptographische Verfahren: Empfehlungen und Schlüssellängen gibt das BSI Empfehlungen, welche Cipher-Suites für TLS eingesetzt werden sollen.

Welche Cipher-Suites aktuell bei einer Webseite eingesetzt werden, kann man ganz einfach unter https://www.ssllabs.com/ssltest testen.



Verwendete TLS Cipher-Suites

All die Gedanken die man sich bei der Auswahl der richtigen Cipher-Suites für TLS 1.2 gemacht hat, muss man sich glücklicherweise bei TLS 1.3 nicht machen. Hier wurden Altlasten schon im Vorherein abgeschnitten, sodass die Konfiguration meistens sehr kurz ausfallen dürfte.

Quellen und Verweise

- https://tools.ietf.org/html/rfc5246
- Sichere Netzwerkkommunikation - Grundlagen, Protokolle und Architekturen; Bless, R., Mink, S., Blaß, E.-O., Conrad, M., Hof, H.-J., Kutzner, K., Schöller, M.
- c't 2 vom 3.01.2022 - Transportverschlüsselungsverwirrung: Sichere Cipher-Suites für TLS auswählen

Artikel vom 30.08.2016