Universität Hamburg

Fachbereich Informatik

Entwurf vom
13. Mai 2015

Exposé

Bachelorarbeit über TLS (1.3)

vorgelegt von

Tom Petersen

geb. am 13. Dezember 1990 in Hannover

Matrikelnummer 6359640

Studiengang Informatik

eingereicht am 13. Mai 2015

1 SSL und TLS - ein Überblick

SSL (Secure Socket Layer) bzw. TLS[1] (Transport Layer Security) ist ein zustandsbehaftetes Protokoll, das auf dem TCP-Protokoll der Transportschicht des TCP/IP-Protokollstapels aufbaut[2]. Es bildet also eine Schicht zwischen Transport- und Anwendungsschicht. Viele Protokolle der Anwendungsschicht nutzen TLS zur sicheren Datenübertragung, so beispielsweise HTTPS oder FTPS.

SSL wurde von der Firma Netscape entwickelt und nachdem es starke Verbreitung gefunden hatte, durch die IETF als TLS 1.0 in RFC 2246 standardisiert (TLS 1.0 entspricht hierbei SSL 3.1). Aktuell ist die TLS-Version 1.2 und an Version 1.3 wird gearbeitet. [Sch09]

Hauptaufgaben von TLS sind Authentifikation der Kommunikationspartner, symmetrische Verschlüsselung der Kommunikation sowie die Sicherstellung der Integrität der übertragenen Nachrichten. Die hierbei verwendeten kryptographischen Verfahren werden erst zu Beginn der Kommunikation festgelegt. [Eck13]

TLS 1.0 RFC 2246 - http://tools.ietf.org/html/rfc2246

TLS 1.1 RFC 4346 - http://tools.ietf.org/html/rfc4346

TLS 1.2 RFC 5246 - http://tools.ietf.org/html/rfc5246

TLS Extensions RFC 3546 - http://tools.ietf.org/html/rfc3546,
RFC 3466 - http://tools.ietf.org/html/rfc4366,
RFC 6066 - http://tools.ietf.org/html/rfc6066

TLS 1.3 Draft - https://tools.ietf.org/html/draft-ietf-tls-tls13-05

1. Im weiteren Verlauf dieser Arbeit wird der Einfachheit lediglich von TLS gesprochen, es ist jedoch ebenso SSL gemeint. Bei etwaigen Unterschieden wird explizit auf diese eingegangen werden.
2. Es gibt auch DTLS (Datagram Transport Layer Security), ein auf TLS basierendes Protokoll, dass auch per UDP Daten übertragen kann

2 Funktionsweise und Teilprotokolle

Zum Einstieg grobe Funktionsbeschreibung, evtl. schon mit Grafik über Verbindungsaufbau?, Grafik der TLS-Protokolle

2.1 Untere Schicht

TLS besteht selbst aus zwei Schichten. In der unteren Schicht befindet sich das *Record-Protokoll*, das die Daten von den Teilprotokollen der oberen Schicht entgegennimmt, diese Anwendungsdaten fragmentiert (maximale Paketgröße 2^{14} Byte) und optional komprimiert. Danach wird je nach aktuell verhandelten krypographischen Funktionen die Integrität der Daten durch Berechnen und Anhängen eines MACs gesichert und die Nachricht verschlüsselt[1].

Nach http://crypto.stackexch schlägt Schneier in Cryptography Engineering MAC-then-encrypt aus Gründen der Komplexheit von Encrypt-then-MAC vor. Nachlesen! Gibt es in der Informatik-Bibliothek: T FER 45399

Der Mac wird auf die folgende Weise berechnet[2]:

$$\begin{aligned} MAC = hash(&MAC\_KEY|pad_2| \\ &hash(MAC\_KEY|pad_1|seq\_num|SSLCompressed.type| \\ &SSLCompressed.length|SSLCompressed.fragment)) \end{aligned}$$

Ich vermute mal, dass das noch das alte Verfahren ist, das in TLS durch HMAC abgelöst wird. Nachschauen, ebenso SSLCompressed vmtl jetzt anders?

wobei $pad_1 = (0x36)\ldots(0x36)$ (48 mal für MD5 und 40 mal für SHA) und $pad_2 = (0x5c)\ldots(0x5c)$ (48 mal für MD5 und 40 mal für SHA) gilt. Diese Werte entsprechen ipad und opad aus dem HMAC-Verfahren ([KBC97]).

Der Record-Header enthält Informationen über die verwendete SSL/TLS-Version, den Content-Type (Handshake, Alert, ChangeCipherSpec, ApplicationData) und die Länge des Klartextfragments.

Aus dem nach Ausführung des Handshake-Protokolls (siehe nächsten Abschitt) beidseitig bekannten Master-Secret werden Schlüssel für die Erstellung des MACs sowie für die Kommunikation zwischen Client und Server berechnet (je einer pro Seite). Dazu werden solange Schlüsselblöcke nach dem folgenden Verfahren erstellt, bis alle Schlüssel konstruiert werden können:

Wahrscheinlich auch das hier ein bisschen anders?

$$\begin{aligned} key\_block = &MD5(ms|SHA('A'|ms|R_C|R_S))| \\ &MD5(ms|SHA('BB'|ms|R_C|R_S))| \\ &MD5(ms|SHA('CCC'|ms|R_C|R_S)) \end{aligned}$$

1. Achtung: hier wird MAC-then-Encrypt angewendet. Laut [BN00] ist Encrypt-then-MAC vorzuziehen.
2. Entnommen aus [Eck13].

2.2 Obere Schicht

In der oberen Schicht sind vier Teilprotokolle spezifiziert: *Handshake-, ChangeCipherSpec-, Alert-* und *ApplicationData-Protokoll*.

Das *Handshake-Protokoll* dient zur Vereinbarung kryptographischer Verfahren und zur Aushandlung eines Schlüssels für die symmetrische Verschlüsselung der später gesendeten Daten. Der Handshake kann folgendermaßen ablaufen (abhängig von optionalem Clientzertifikat, Preshared Key, erneute Verbindung, ...; entnommen aus [Eck13]):

als ordentliche Grafik ohne Clientverifikation

- –> client_hello mit
  $H_C = (4$ Byte Zeitstempel, 28 Byte Zufallszahl, Sitzungsidentifikator (!= Null bei bereits existierender Sitzung), Cipher Suite$)$
- <– server_hello mit
  $H_S = (4$ Byte Zeitstempel, 28 Byte Zufallszahl, Sitzungsidentifikator (falls vom Client gewünschte Sitzung bedient werden kann), Cipher Suite, die vom Server unterstützt wird$)$
- <– Server Zertifikat (inklusive öffentlichem Schlüssel des Servers, meist nach X.509v3)
- Client: Zertifikatverifikation
- –> Generierung und Senden von PreMasterSecret (48 Byte) verschlüsselt mit öffent. Schlüssel des Servers (bei RSA) oder Diffie-Hellman-Verfahren
- Client und Server: aus Zufallszahl des Clients R_C aus H_C, Zufallszahl des Servers R_S aus H_S und PreMasterSecret wird das MasterSecret berechnet

so wahrscheinlich < TLS 1.2, danach Pseudozufallsfunktion -> Nachschauen

$$\begin{aligned} master\_secret = & MD5(Pre|SHA('A'|Pre|R_C|R_S))| \\ & MD5(Pre|SHA('BB'|Pre|R_C|R_S))| \\ & MD5(Pre|SHA('CCC'|Pre|R_C|R_S)) \end{aligned}$$

- –> Handshakeabschluss mit MD5 und SHA-MAC über alle bisher ausgetauschten Nachrichten
- <– Handshakeabschluss mit MD5 und SHA-MAC über alle bisher ausgetauschten Nachrichten

Das *ChangeCipherSpec-Protokoll* dient dazu, die vereinbarten kryptographischen Verfahren zu ändern. Es enthält lediglich eine Nachricht mit dem Wert 1, die für das Übernehmen der während des Handshakes ausgehandelten Verfahren steht. [Eck13]

Das *Alert-Protokoll* dient dazu, auftretende Fehler oder Warnungen zu versenden, die während des Datenaustausches auftreten.

Übersicht

Das *ApplicationData-Protokoll* ist zuständig für das Durchreichen von Anwendungsdaten, die von der Anwendungsschicht gesendet werden sollen.[Sch06]

2.3 Sitzungs- und Verbindungskonzept

TLS erstellt beim ersten Handshake eine Sitzung zwischen Client und Server. Hierbei wird ein Sitzungsidentifikator erstellt, der beim server_hello mitgesendet wird. Weiterhin wird sich in der Sitzung das Zertifikat des Gegenübers, optional das Kompressionsverfahren, die CipherSpec und das MasterSecret gemerkt.

Ein Client kann nun, wenn er den Sitzungsidentifikator beim client_hello mitschickt, eine alte Sitzung in Form einer neuen Verbindung wiederaufnehmen oder mehrere Verbindungen parallel aufbauen. Eine Verbindung wird dabei durch Client- und Server-Zufallszahlen R_C und R_S, die generierten Schlüssel, einen Initialisierungsvektor sowie aktuelle Sequenznummern beschrieben.

Wie wird sich auf den geeinigt? Übertragen? Aus mastersecret?

Beim Verbindungsaufbau kann so ein verkürzter Handshake genutzt werden, bei dem weniger Nachrichten gesendet werden müssen. Es kann dabei auf Neuberechnung des master-secrets, Server- und Client-Validierung und Aushandlung der CipherSpec verzichtet werden.

Grafik des verkürzten Handhakes

2.4 Ciphersuites

3 Angriffe

3.1

Literaturverzeichnis

[BN00] Mihir Bellare und Chanathip Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In *Advances in Cryptology - ASIACRYPT 2000*. Springer Berlin Heidelberg, 2000.

[Eck13] Claudia Eckert. *IT-Sicherheit - Konzepte, Verfahren, Protokolle (8. Aufl.)*. Oldenbourg Verlag, München, 2013.

[KBC97] H. Krawczyk, M. Bellare, und R. Canetti. Hmac: Keyed-hashing for message authentication. RFC 2104, 1997.

[Sch06] Bruce Schneier. *Angewandte Kryptographie - Der Klassiker. Protokolle, Algorithmen und Sourcecode in C*. Pearson Studium, München, 2006.

[Sch09] Klaus Schmeh. *Kryptographie - Verfahren, Protokolle, Infrastrukturen*. dpunkt.verlag, Heidelberg, 2009.