

SSL

Algorithmen und Anwendung

Stefan Pfab
sisspfab@stud.uni-erlangen.de

Abstract

Viele Anwendungen erfordern nicht nur eine eindeutige und zuverlässige Identifizierung der an einer Kommunikation beteiligten Partner, sondern zusätzlich die Bereitstellung eines sicheren Kommunikationskanals zwischen diesen Partnern. Unter dieser Zielsetzung wurde SSL als Ergänzung zum weit verbreiteten TCP/IP-Protokoll entwickelt.

1 Systemarchitektur

1.1 Entwicklung von SSL

Die Entwicklung von SSL geht zurück ins Jahr 1994. Damals veröffentlichte Netscape erste Spezifikationen zu einer neuen Protokollschicht. Sie ist nach ISO/OSI zwischen der Anwendungsschicht und einer zuverlässigen Übermittlungsschicht angesiedelt und sollte im wesentlichen zwei Aufgaben erfüllen:

- gegenseitige Authentifizierung von zwei Kommunikationspartnern
- Verschlüsselung der Kommunikation zwischen den Partnern

Wesentliche Konzepte sind die Verwendung sowohl symmetrischer als auch asymmetrischer Verschlüsselungsverfahren und die Benutzung von Zertifikaten zur Authentifizierung. Anfang 1995 brachte Netscape eine fertig implementierte Version ihrer ersten Entwürfe als „SSL V2“ auf den Markt. Der zu diesem Zeitpunkt aktuelle Navigator 3.0 enthielt eine clientseitige Ergänzung des TCP/IP-Protokollstacks. Serverseitig war es der Netscape Internet Server, später ergänzt um das Gespann Apache mit einem SSL-Paket wie `ssleay` oder `openssl`. Ende 1995 erschien die Version V3, die im wesentlichen einige Bugs aus der Version 2 behob und außerdem neue Kryptographiealgorithmen beinhaltete. Diese Version ist bis zum heutigen Tage aktuell, obwohl es Bemühungen von seiten der IETF (Internet Engineering Task Force) gibt, SSL V3 zu einem neuen Standard namens TLS (Transport Layer Security) zu erweitern. Dieser wurde aber noch nicht verabschiedet, er ist folglich auch noch in keinem Programm zu finden. Nachdem außerdem keine nennenswerten und erfolgversprechenden Angriffe bekannt sind (s.u.), besteht hierzu auch kein großer Zeitdruck.

1.2 Protokollstack

Der Protokollentwurf für SSL sieht vor, dass zwischen die Anwendungsschicht und die Übermittlungsschicht des IOS/OSI-Schichtenmodells eine neue Instanz geschoben wird, die einen

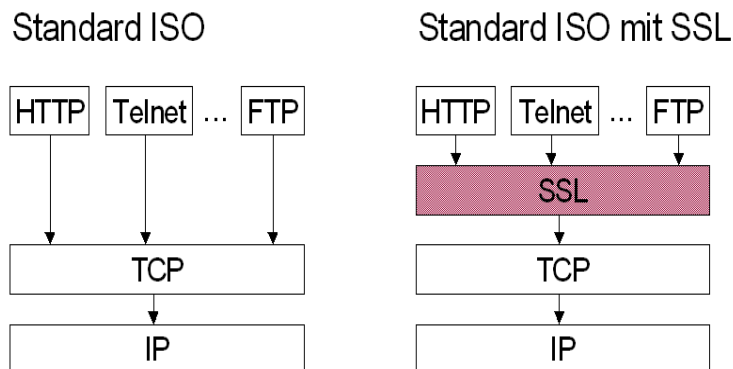


Abbildung 1: Vergleich ISO-Protokollstack mit und ohne SSL

Dienst zur Ver- und Entschlüsselung eines Datenstromes und zur Authentifizierung des Kommunikationspartners anbietet.

Dieses Vorgehen bietet zwar den wesentlichen Vorteil, für ein breites Spektrum an Anwendungen zugänglich zu sein, jedoch macht dies Anpassungen bei den jeweiligen Programmen notwendig. Während für eine normale Netzwerkverbindung in C ein

```
new_socket = socket(...);
connect(new_socket,...);
...
char *message = Hello World!;
write(new_socket, message, strlen(message));
```

genügt, muß für die Verwendung von SSL ein höherer Aufwand betrieben werden[8]:

```
new_socket = socket();
connect(new_socket , );
SSL_set_fd(ssl, new_socket );
SSL_connect(ssl);
server_cert = SSL_get_peer_certificate(ssl);
char *message = Hello World!;
SSL_write(new_socket , message, strlen(message));
```

1.3 Verbindungsaufbau

Der Verbindungsaufbau bei SSL wird üblicherweise vom Client initiiert. Dabei fordert er einen Server auf, seine Identität mittels eines Zertifikates zu beweisen und dabei seinen Public Key zu übertragen. Der Client berechnet aus einem zusätzlich zum Zertifikat übertragenen Zufallswert und einigen weiteren Hashwerten aus dem Zertifikat den symmetrischen Schlüssel für die Verschlüsselung des folgenden Datentransfers. Der Client überträgt den symmetrischen Schlüssel, verschlüsselt mit dem Public Key an den Server, wodurch beide Kommunikationspartner im Besitz des selben geheimen Schlüssels sind. Ab diesem Zeitpunkt findet der Datentransfer verschlüsselt mittels eines symmetrischen Verfahrens statt. Der konkret verwendete Verschlüsselungsalgorithmus wird beim Aufbau ausgehandelt, in Abhängigkeit der Version von SSL und den verfügbaren Algorithmen. SSL V3 sieht als asymmetrische Verfahren RSA und Diffie-Hellmann vor, als symmetrische DES, 3DES, Idea und RC4 (ein symmetrisches Verfahren von RSA).

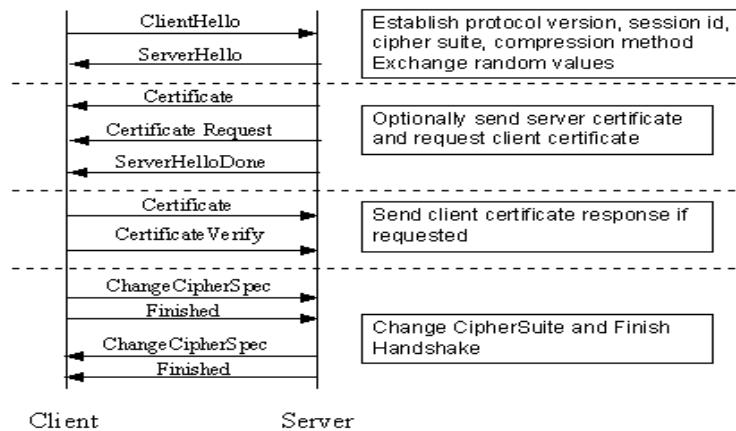


Abbildung 2: Aufbau einer SSL-Verbindung

Ab dem Zeitpunkt, der in Abb. 2 als „Finished“ bezeichnet ist, findet die gesamte Kommunikation zwischen Client und Server verschlüsselt mittels eines symmetrischen Verfahrens statt. Damit werden die Vorteile der beiden Verschlüsselungsverfahren, symmetrischer und asymmetrischer, in idealer Weise kombiniert:

- Der problemlose Schlüsselaustausch der asymmetrischen Verfahren über unsichere Kanäle vereinfacht den Verbindungsaufbau zwischen beliebigen Kommunikationspartnern.
- Die symmetrischen Verfahren bieten eine hohe Performanz, sodass beim Datentransfer ein hoher Durchsatz gesichert ist.

Zusätzlich zu diesen Möglichkeiten bietet das SSL-Protokoll eine Vielzahl an Ergänzungen, um die Sicherheit der Datenübertragung zu erhöhen, beispielsweise Authentifizierung des Client gegenüber des Servers durch ein Client-Zertifikat, regelmäßiger Schlüsselwechsel während einer Session oder Erhöhung der Schlüssellänge für das symmetrische Verfahren.

2 Zertifikate

Zur Authentifizierung des Kommunikationspartners muss neben der sicheren Übermittlung der Identität der Gegenstelle auch gesichert sein, dass die Gegenstelle auch wirklich diejenige ist, für die sie sich ausgibt. Dies wird im SSL-Protokoll durch die Verwendung von Zertifikaten nach X.509-Standard erreicht. Abbildung 3 verdeutlicht deren Aufbau.

Der Aufbau dieser Zertifikate leistet für SSL zwei wesentliche Dienste: das Zertifikat beinhaltet den Public Key des Zertifikat-Inhabers und es enthält den Aussteller des Zertifikates, die sog. Certification Authority, kurz CA.

2.1 Certification Authorities

Zur sicheren Validierung der Identität einer Einrichtung oder Person ist i.a. eine dritte, unabhängige Stelle nötig, der beide Kommunikationspartner vertrauen. Für X.509-Zertifikate sind dies sog. Certification Authorities (CA), z.B. die Firmen VeriSign oder Nortel Networks. Derjenige, der die Erstellung eines beglaubigten Zertifikates wünscht, muss bei einer dieser

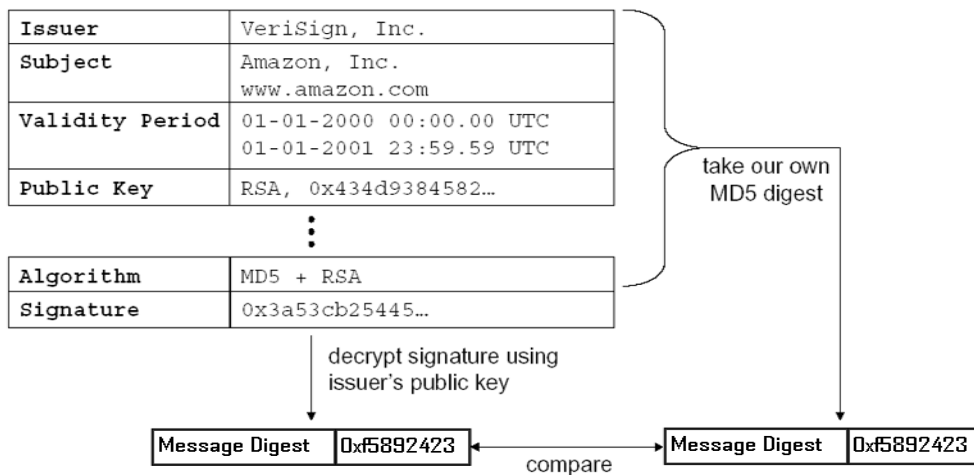


Abbildung 3: Aufbau eines X.509-Zertifikats

CAs einen Antrag für ein Zertifikat stellen. Die Certification Authority überzeugt sich von der Identität des Antragstellers auf „konventionellem“ Weg z.B. über eine beglaubigte Kopie des Personalausweises oder nützt das in Deutschland mögliche Verfahren des PostIdent[7]. Zusätzlich wird das eigene Zertifikat von der CA signiert, d.h. weitere, verschlüsselt gespeicherte Informationen über die CA werden dem Zertifikat hinzugefügt. Danach ist es jedem Benutzer selbst überlassen, welcher CA er vertraut und welcher nicht...

Für den Gebrauch in Intranets oder anderen kleinen Umgebungen ist es selbstverständlich möglich, ein sog. „Self-Signed Certificate“ zu erzeugen, bei dem die CA und der Inhaber dieselbe Person sind. Dies ist oftmals sinnvoll, da die meist kommerziellen Certification Authorities ihre Dienste nur gegen Bezahlung anbieten. So kostet bei VeriSign ein SSL-Zertifikat für eine Privatperson mit einer Gültigkeit von einem Jahr 895\$[6].

3 Sicherheit

Um die Sicherheit von SSL zu bewerten, müssen zwei Angriffsszenarien unterschieden werden: der Angriff auf die verwendete Kryptographie und der Angriff auf das Protokoll selbst. Angriffe auf die Kryptographie selbst (Mitlesen der übertragenen Daten und anschließende Entschlüsselung) sind nach heutigem Stand zwecklos (siehe Vorträge über symmetrische und asymmetrische Verfahren). Das Verfahren des Verbindungsaufbaus dagegen bietet einige Angriffspunkte, entweder in den Besitz des symmetrischen Schlüssels zu gelangen und so passiv den Traffic mitzulesen oder vollständig den eigenen Host dem Client als Server unterzuschieben. Diese Angriffe erfordern aber sehr weitgehende technische Möglichkeiten und ein umfangreiches Wissen um das Umfeld des Ziels (z.B. die Möglichkeit, unmittelbar einzelne TCP-Pakete abzufangen, zu modifizieren oder stattdessen eigene Pakete zu senden). Zu den einzelnen Angriffsmöglichkeiten verweise ich auf folgende Quelle: [3].

Literatur

- [1] <http://www.openssl.org/>
- [2] <http://citeseer.nj.nec.com/252522.html>
- [3] <http://citeseer.nj.nec.com/mitchell98finitestate.html>
- [4] <http://www.phaos.com/sslresource.html>
- [5] <http://www.pseudonym.org/ssl/>
- [6] <http://www.verisign.com/products/site/secure/index.html>
- [7] <http://www.deutschepost.de/brief/js/index.html?/brief/produkte-services/postident/inhalt.html>
- [8] <http://www2.psy.uq.edu.au/ftp/Crypto/ssl.html>