Ausarbeitung

# SSL, robuster Protokollentwurf und Angriffe

Stud. Inf. Doris Diedrich

Lehrstuhl für Kryptographie & Sicherheit
Prof. Dr. Birgit Pfitzmann
Fachbereich Informatik
Universität Saarbrücken

Im Sommersemester 2001

**Zusammenfassung**

Im der Ausarbeitung zum Vortrag „Angriffe auf SSL“werden verschiedene Angriffe auf verschiedene Versionen von SSL erläutert. Dazu wird zunächst noch einmal auf die wichtigsten Aspekte von SSL eingegangen um darauf aufbauend SSL unter dem Aspekt des robusten Protokollentwurfs näher zu betrachten.

Die Probleme, die bei der Betrachtung von SSL unter dem Aspekt des robusten Protokollentwurfs deutlich werden, führen zu den Schwachstellen von und Angriffen auf SSL, die im letzten Abschnitt erlütert werden.

## Inhaltsverzeichnis

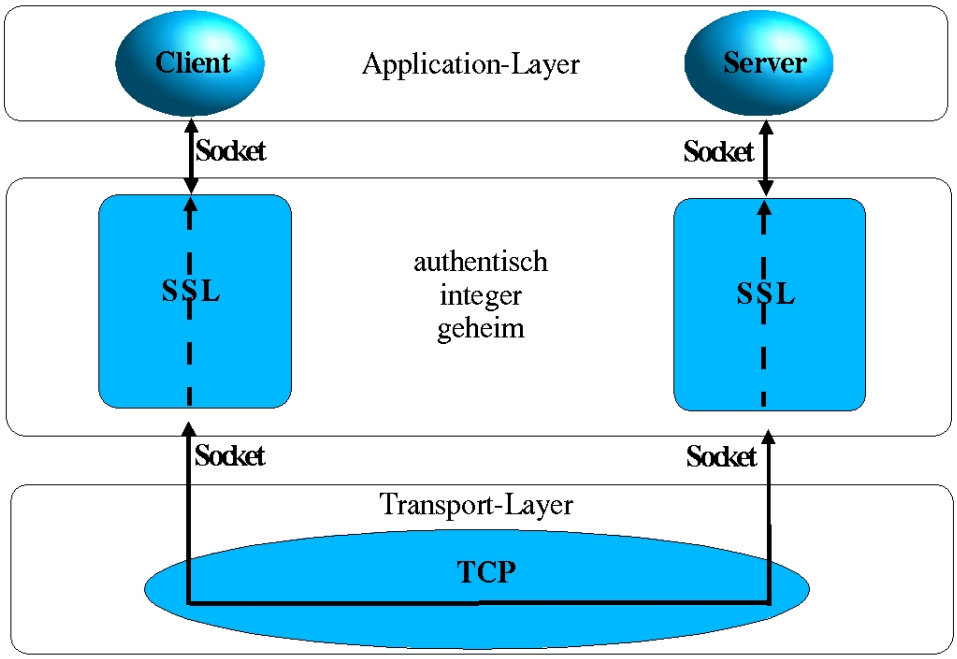


Abbildung 1: Die SSL-Schicht schiebt sich transparent zwischen den Client und den Transport-Layer.

# 1 Grundlagen: Aufbau von SSL

SSL ist ein Protokoll, das benutzt wird um Kommunikationen zwischen zwei Parteien zu sichern. Die Kommunikation soll geheim und integer ablaufen. Auf Wunsch der Kommunikationsteilnehmer soll auch die Authentizität der Gesprächspartner sicher gestellt sein. (Ich möchte meine Bankdaten nur meinem Onlinebuchhändler verraten.)

Wird von einer Appikation eine SSL-Verbindung gewünscht, so baut sie wie üblich eine Socket-Verbindung auf. Anstatt jedoch direkt den Transport-Layer anzusprechen wird stattdessen, wie in Abbildung 1 zu sehen ist, ein SSL-Programm aufgerufen.

Dieses stellt nach oben hin einen Socket zur Verfügung, so dass die neue Sicherungsschicht für die aufrufende Applikation weitgehend transparent benutzbar ist.

Nach unten hin baut SSL zunächst wie üblich über TCP eine Verbindung auf. Anstatt aber sofort mit dem Datenaustausch zu beginnen, baut das SSL-Programm zunächst einen sicheren Kanal zum Kommunikationspartner auf. Erst danach werden Daten der Applikation, zB ein HTTP-Request, an den Kommunikationspartner weiter geleitet.

## 1.1 Interaktion der Protokolle innerhalb SSL

Einen berblick über die verschiedenen Protokolle, die innerhalb SSL arbeiten, erhält man in Abbildung 2. Dort sieht man auch, wie die Protokolle miteinander und mit der Applikation interagieren.

Der Ablauf einer Verbindung ist wie folgt: Zunächst tritt das Handshake-Protokoll in Aktion und handelt die zur Kommunikation notwendigen Parameter, zB den

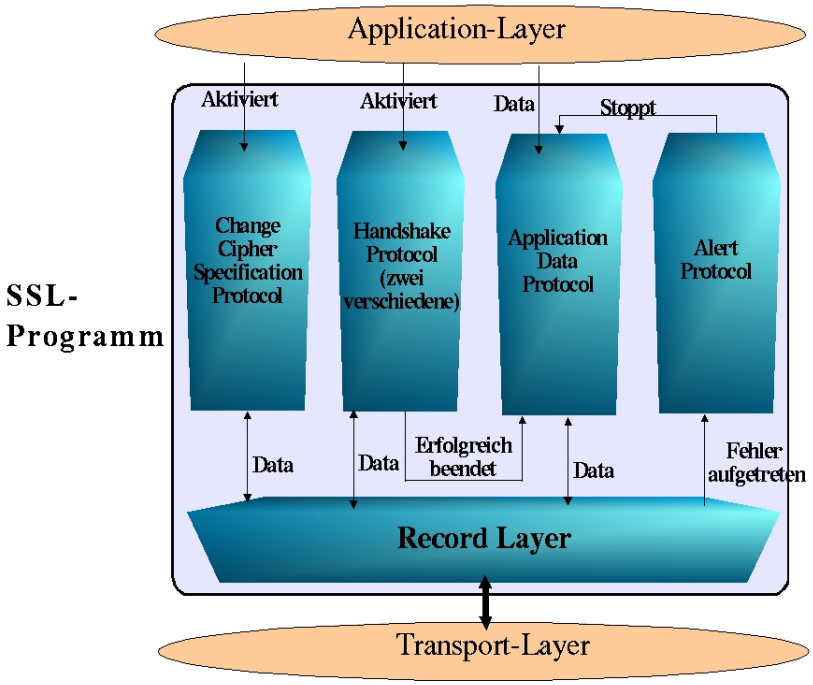


Abbildung 2: Aufbau von SSL mit den verschiedenen Layern und Protokollen.

Verschlüsselungsalgorithmus, aus. Gegebenenfalls werden auch Zertifikate ausgetauscht, die für die Authentizität der Gesprächspartner sorgen sollen. Näheres zu den ausgetauschten Nachrichten und Parametern s.u.

Sind die Parameter ausgehandelt, übernimmt das Application-Data-Protokoll die Kommunikation, das einfach nur Daten der Applikation zum Record-Layer durchreicht. Entdeckt der Record-Layer einen Fehler, so wird dieser dem Alert-Protokoll gemeldet, das die Fehlerbehandlung bestimmt und unter Umständen sogar die Kommunikation abbricht.

Das Alert-Protokoll ist auch für den Abbruch der Kommunikation zuständig, wenn kein Fehler auftritt, nachdem es also von der Applikation über den gewünschten Abbruch der Verbindung informiert wurde.

Das Change-Cipher-Spec-Protokoll dient nur dazu, das Aushandeln neuer Parameter einzuleiten. Es besteht nur aus einer Nachricht, der Change-Cipher-Spec-Nachricht.

## 1.2 Protokolle

### 1.2.1 Record-Layer

Sofort nach dem Aufruf des SSL-Programmes tritt der Record-Layer in Funktion. Dieser arbeitet, da ihm noch keine Parameter wie Schlüssel oder Verschlüsselungsalgorithmen bekannt sind, in einem 0-Modus: er ist zwar aktiv, verschlüsselt und berechnet den MAC[1] aber sozusagen mit 0-Werten.

Alle zwischen den Kommunikationspartnern ausgetauschten Daten und Nachrichten werden von den beiden Record-Layern der Kommunikationspartner bearbeitet.

[1] Message Authentication Code, hier durch Hashen realisiert.

Im Einzelnen fragmentiert der Record-Layer die Daten zunächst zu Paketen einheitlicher Gröe. Dann berechnet er für jedes Paket einen MAC und verschlüsselt zuletzt jedes Paket einzeln. Der Record-Layer behandelt dabei alle ihm übergebenen Daten gleich, völlig unabhängig davon ob es sich um Nachrichten von SSL selbst oder um Anwendungsdaten aus der Applikation handelt.

### 1.2.2 Das Handshake-Protokoll

In Abbildung 3 sieht man den Aufbau der wichtigsten Nachrichten des Handshake-Protokolls. Sie sind zu Beginn der Kommunikation noch kaum gesichert, da sie ausgetauscht werden, bevor Schlüssel bekannt sind. Erst mit dem Austausch der Finished-Nachrichten werden sie authentifiziert.

Parameter wie die verwendete SSL-Version und der Schlüsselaustauschalgoritmus sowie der Verschlüsselungsalgorithmus müssen erst zwischen den Kommunikationspartnern abgesprochen werden. Dazu dient ein Teil der Client-Hello-Nachricht: die Liste von Cipher-Suites. Eine Cipher-Suite ist nichts anderes als eine Menge von Algorithmen: Ein Schlüsselaustauschalgorithmus, ein Algorithmus zum Signieren, ein MAC-Berechnungs-Algorithmus und ein Verschlüsselungsverfahren. Ein Name einer solchen Cipher-Suite ist zB
SSL_DH_RSA_EXPORT_WITH_DES40_CBC_SHA. Das bedeutet: SSL soll DH-Schlüsselaustausch machen, RSA-Zertifikate benutzen, den Exportschlüssel 40 und DES als Verschlüsselungsalgorithmus verwenden. Als Betriebsart wird CBC gewünscht, SHA ist der gewünschte Hash-Algorithums[2]. Initial arbeitet der Record Layer mit der Cipher-Suite
SSL_NULL_WITH_NULL_NULL.

Die von SSL verwendeten Verschlüsselungsalgorithmen sind vom Implementierer fast beliebig zu ergänzen, so dass neu hinzukommende Algorithmen modular in SSL-Implementierungen eingebracht werden können. Es müssen dann nur neue Namen für Cipher-Suites gefunden und abgesprochen werden.

# 2 SSL und Robuster Protokollentwurf

Etwa zur gleichen Zeit wie SSL wurden Regeln entwickelt, deren Beachtung beim Protokollentwurf zu weniger angreifbaren, also robusteren Protokollen führen soll.

Entsprechend sind bei der Entwicklung von SSL noch nicht alle Regeln zum robusten Protokollentwurf beachtet worden. Das führte zu einigen Schwachstellen und Angriffspunkten im Protokoll auf die im Folgenden noch näher eingegangen wird.

## 2.1 Sequenznumbers und Nonces

SSL unterstützt als Freshness-Manahmen Sequenznumbers und Nonces, es verwendet jedoch keine Zeitstempel. Wie im Script[4] ausgeführt, sind Zeitstempel nur für geschlossene Umgebungen mit bestimmten Eigenschaften sinnvoll. SSL wurde für das Internet entwickelt, wo die Verwendung von Zeitstempeln problematisch ist.

Sequenznummern werden im Record-Layer zugefügt, nachdem fragmentiert wurde und bevor der MAC berechnet wird. (D.h.: die Sequenznummer wird mit gehasht.) Sequenznummern verhindern Replay-Angriffe innerhalb einer Sitzung. Allerdings ist der Test der Sequenznummern nicht spezifiziert.

[2] Es kann nur zwischen MD5 und SHA gewählt werden. Beide Algoritmen werden jedoch ausser zur Authentikation der Nachrichten noch fest verdrahtet zur Schlüsselerzeugung und zu anderen Zwecken benutzt. Der Benutzer hat darauf keinen echten Einfluss.

Die Art, wie in SSL Nonces verwendet werden, bietet eine gewisse Sicherheit gegen die Wiederaufnahme alter Sitzungen mit denselben Schlüsseln: Als Nonces dienen die in der Handshake-Nachricht ausgetauschten Client und Server Random, also Zufallswerte. Sowohl Client als auch Server erzeugen je einen Random. Beide werden zur Erzeugung aller benötigten Sitzungsschlüssel verwendet. Das bietet für beide Seiten eine gewisse Freshness-Garantie, die bei Verwendung einer einzigen Nonce, die entweder vom Client oder vom Server erzeugt wurde, nur für eine Seite gelten würde.

## 2.2 TransaktionsID: SessionId?

SSL erzeugt für eine Session (eine Kommunikationssitzung) eine SessionId, die zu Beginn der Kommunikation mit übertragen wird. Man kann sie jedoch nicht als TransaktionsID im Sinne des robusten Protokollentwurfs bezeichnen.

Der eigentliche Zweck der SessionId in SSL ist, alte Sessionparameter aufzubewahren, damit diese für eine neue Session nicht erneut ausgehandelt werden müssen.

Für Freshness und Replayvermeidung sind eher die Nonces und die Sequenznummern verantwortlich.

## 2.3 Explizitheit

SSL gibt für die verwendeten Teil-Protokolle (z.B. das Handshake-Protokoll) feste Nachrichtenformate vor. Tests auf die Korrektheit der Formate sind jedoch nicht für alle ausgetauschten Nachrichten vorgeschrieben. Es bleibt dem Programmierer überlassen, welche Bedingungen er für die Korrektheit der empfangenen Nachrichten stellt. Das gilt besonders für die Nachrichten im Handshake Protokoll und für die Zertifikate.

Problematisch ist dabei vor allem, dass speziell im Handshake-Protokoll die Nachrichten zunächst unverschlüsselt übertragen werden müssen. Anders als in anderen Protokollen, wie z.B. IPsec, bei denen Schlüssel vorab ausgetauscht werden können, ist bei SSL im Allgemeinen zu Beginn einer Kommunikation vom angesprochenen Server noch kein Schlüssel bekannt[3]. Die Kommunikation ist also bis zu dem Zeitpunkt unverschlüsselt, bis zu dem Schlüssel erfolgreich ausgetauscht worden sind.

SSL sichert die Handshake-Nachrichten erst durch Finished-Nachrichten, die nach Abschluss des Schlüsselaustausches durch einen MAC gesichert und verschlüsselt übertragen werden. Siehe Abbildung 3. Sie enthalten den Hashwert über alle Handshake-Nachrichten und authentifizieren die im Handshake-Protokoll ausgetauschten Nachrichten.

Wenn es einem Man-In-The-Middle gelingt, seinen Angriff durchzuführen, bevor die Finished-Nachrichten ausgetauscht wurden, kann er die Finished-Nachrichten fälschen und der Angriff bleibt unentdeckt.

## 2.4 Zertifikate

Ebenfalls ein Problem der mangelnden Explizitheit der Spezifikation ist, dass keine Tests für die Zertifikate vorgesehen sind.

Die Formate der verschiedenen Zertifikate, die unterstützt werden, sind bis ins Detail spezifiziert. Keine Aussage allerdings wird darüber gemacht, wie und nach welchen Kriterien die Zertifikate geprüft werden sollen, zB auf Gültigkeit.

Das führte beim Internet-Explorer zu einem Angriffspunkt: In gewissen Versionen des IE wurde zwar die Gültigkeit eines Zertifikates geprüft, es wurde jedoch

[3]Das ist so, weil SSL dafür ausgelegt ist, auch unbekannte Server ansprechen zu können.

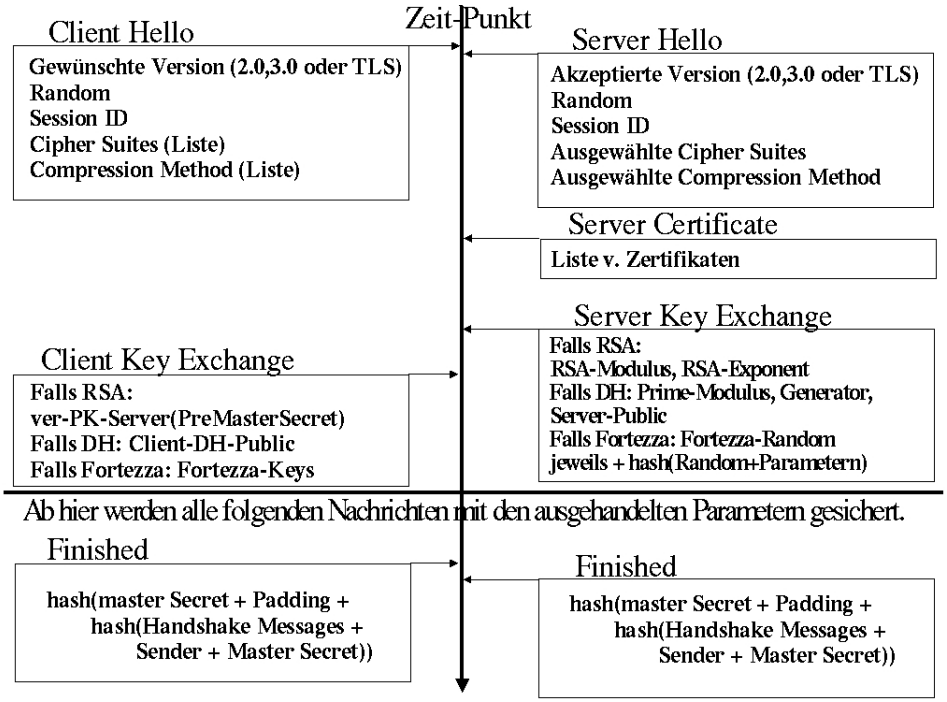


Abbildung 3: Aufbau einiger Nachrichten innerhalb des Handshake-Protokolls. Man sieht, dass viele Felder bis zum Austausch der Finished-Nachrichten ungeschützt bleiben.

nicht überprüft, ob das gültige Zertifikat auch zum Absender gehörte. Ein Man-In-The-Middle konnte also den IE überlisten, indem er sein eigenes, gültiges Zertifikat schickte.

### 2.5 Dispute

SSL unterstützt keine Dispute. Diese sind im Entwurf nicht vorgesehen und vermutlich auch nicht sinnvoll, da SSL unterhalb der Applikationsschicht transparent arbeitet. Dispute sollten innerhalb der Applikation behandelt werden.

## 3 Angriffe

Wie bereits erläutert, liegen die Haupt-Angriffspunkte des SSL Protokolls in den Nachrichten des verhältnismässig schwierig zu schützenden Handshake-Protokolls[4]. In [5] sind weitere Schwachstellen von SSLv3.0[2], im Record Layer und in anderen Teilprotokollen, aufgeführt. Da diese, soweit bekannt, nicht zu konkreten Angriffen geführt haben, beschränke ich mich hier auf die Angriffspunkte im Handshake-Protokoll.

SSLv2.0[1] gilt generell als recht unsicheres Protokoll, das nicht mehr verwendet werden sollte. Näheres dazu kann man auch in [5] finden. Hier wird nur der Cipher-Suite-Rollback-Angriff geschildert, der wohl ausschlaggebend für die letztendliche Umstellung auf die SSLv3.0 war.

Zunächst aber einige Angriffe, die oft fälschlich als Angriffe auf SSL bezeichnet werden, es eigentlich aber nicht sind.

[4] Die Handshake-Nachrichten können noch nicht verschlüsselt werden, da zu Beginn des Kommunikationsaufbaus noch keine Verschlüsselungsparameter ausgehandelt sind.

## 3.1 Der Bleichenbacher-Angriff

Der Bleichenbacher-Angriff[6] ist darauf ausgerichtet, den geheimen Schlüssel eines Servers mittels einer Schwachstelle in der PKCS#1 zu finden. Er ist also eigentlich kein Angriff auf SSL sondern auf eine PKCS. Der Angriff nutzt aber die Orakel-Eigenschaften[5] von SSL aus. Speziell verwendet er die Client-Exchange-Nachricht, in der das Pre-Master-Secret mit dem Public-Key des Server verschlüsselt wird.

Ein Angreifer hat das Ziel, Informationen über seine Chosen-Ciphertexte zu gewinnen. Dazu schickt er dem Server ein Client-Hello und danach eine Client-Key-Exchange-Nachricht.

Diese enthält statt dem korrekt verschlüsselten Pre-Master-Secret den Chosen-Ciphertext des Angreifers. Der Server versucht, den Chosen-Ciphertext zu entschlüsseln: findet er eine korrekte PKCS-Form vor, dann erwartet er die nächste Nachricht des Client und antwortet korrekt. Klappt die Entschlüsselung nicht, gibt der Server eine Fehlermeldung aus oder bricht die Kommunikation einfach ab. In jedem Fall hat der Angreifer Informationen über seinen Chosen-Ciphertext gewonnen.

Sind so genügend Chosen-Ciphertexte bearbeitet worden, dann kann der Angreifer einen Angriffspunkt in der PKCS#1 ausnutzen und den Secret-Key des Servers bestimmen.

## 3.2 Angriffe auf die Applikation

Es existieren einige Angriffe auf Applikationen, die SSL benutzen. Diese Angriffe werden oft fälschlich als Angriffe auf SSL bezeichnet. Zum Teil sind es Angriffe, die inkorrekte Implementierungen von SSL ausnutzen. ZB gab es einen Angriff der die Verwendung eines schlechten Zufallszahlengenerators ausnutzte.

Aber es existieren auch Angriffe auf Benutzungsoberflächen, die es, zB im Fall von IE oder Netscape, oft zulassen, dass unaufmerksame Benutzer über den Ursprung der gelieferten Webseiten getäuscht werden.

### 3.2.1 Web-Spoofing

Ein Man-In-The-Middle schickt einem Benutzer einen gefälschten Link. Er könnte zB wie in Abbildung 4 gezeigt, O gegen 0 austauschen oder ähnliche optische Tricks anwenden. Um den Angriff noch echter zu gestalten, kann der Man-In-The-Middle sogar die originalen Webseiten verwenden, deren Links er mit solchen auf seinen eigenen Server vertauscht.

Ein so über den Kommunikationspartner getäuschter Anwender wird unter Umständen geheime Daten dem Man-In-The-Middel liefern statt dem vermeintlich angesprochenen Webserver.

### 3.2.2 Falsche Zertifikate

Ein ähnlicher Angriff lenkt die SSL-Kommunikation zwischen Client und Server über den Man-In-The-Middle um. Damit der Client über den tatsächlichen Kommunikationspartner getäuscht werden kann, tauscht der Man-In-The-Middel das Original-Server-Zertifikat gegen sein eigenes aus. Akzeptiert die Applikation gültige Zertifikate, ohne die bereinstimmung des Zertifikatbesitzers mit dem angesprochenen Server zu akzeptieren[6], oder wird der Benutzer nicht deutlich auf die Gefahr hingewiesen, dann ist der Angriff erfolgreich.

[5] Ein Orakel im kryptographischen Sinne ist eine Instanz, die Informationen zu Ciphertexten liefert.

[6] So geschehen in älteren Versionen des IE.

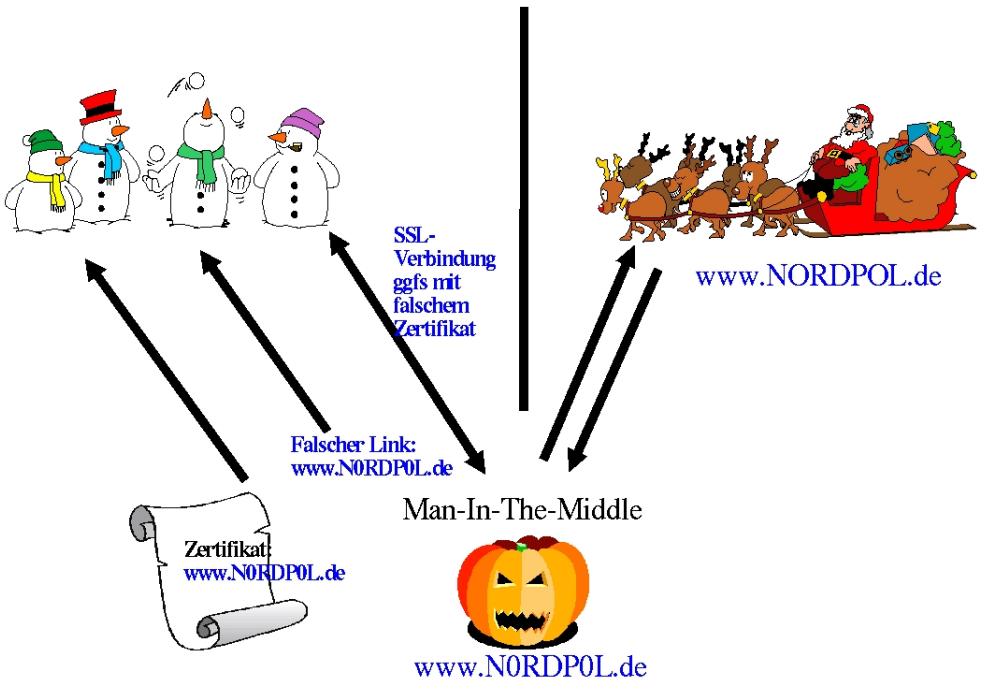


Abbildung 4: Ein Man-In-The-Middle lenkt den Kommunikationsflu um.

### 3.3 Ciphersuite Rollback-Angriff

Dabei handelt es sich um einen Angriff auf SSLv2.0 der in [5] erläutert wird. In SSLv2.0 wird die Integrität der Handshake-Nachrichten kaum geschützt. Anders als in SSLv3.0 gibt es hier keine Finished-Nachricht, die den Inhalt der Handshake-Nachrichten nachträglich sichert.

Ein Man-In-The-Middel kann also die vom Client erzeugte, authentische Liste von Ciphersuites, gegen eine von ihm selbst erzeugte Liste austauschen. Z.B. könnte der Angreifer die Cipher-Suite mit dem vom Client bevorzugten IDEA-Verschlüsselungsverfahren austauschen gegen eine für ihn angenehmere Cipher-Suite[7].

Der Austausch der Cipher-Suite-Liste durch den Angreifer verursacht keine Fehlermeldung und wenn die Benutzungsoberfläche keine Rückmeldung über das verwendete Verfahren gibt, bleibt der Angriff unentdeckt.

## 4 Angriffe auf SSLv3.0

### 4.1 Key-Exchange-Algorithm-Rollback[5]

Dieser Angriff wurde benannt in Anlehnung an den Ciphersuite-Rollback-Angriff, obwohl hier kein Rollback stattfindet, sondern eher ein Austausch eines Algorithmus gegen einen anderen.

Die Spezifikation sagt nichts darüber aus, wie in den Handshake-Nachrichten das Feld mit dem gewünschten Schlüsselaustausch-Algorithmus authentifiziert werden soll. Erst mit der Finished-Nachricht werden die Key-Exchange-Nachrichten zusammen mit allen anderen Handshake-Nachrichten authentifiziert.

[7] Zur Erinnerung: der Client schickt dem Server eine Liste mit den von ihm unterstützten Ciphersuites, und zwar in der Reihenfolge, wie sie von ihm präferiert werden. Siehe dazu auch Abbildung 3. Der Server sucht sich aus der Liste die erste Cipher-Suite aus, die er ebenfalls unterstützt.

Ein Man-In-The-Middle fälscht die vom Client geschickte Hello-Nachricht mit der Liste von Cipher-Suites und trägt Diffie-Hellmann als gewünschten Key-Exchange-Algorithmus ein. Danach fälscht er die Antwort des Servers, so dass der Client weiterhin glaubt, es soll RSA verwendet werden.

Sowohl Client als auch Server wählen sich Key-Exchange-Algorithmus-Parameter. Der Server glaubt an DH und wählt sich also einen primen Modulus p, einen Generator g und einen Server-Public und schickt diesen an den Client. Der Client antwortet mit einer Nachricht, die das, mit diesen Werten allerdings RSA-verschlüsselte, Pre-Master-Secret enthält. Das heisst: der Client verwendet den ersten Parameter p, also die Primzahl, die ihm vom Server geschickt wird, als Modulus, und den zweiten, g, als Exponenten. Er berechnet also:

$m^g modulo p$.

Der Angreifer fängt die Keyexchange-Nachrichten ab und berechnet seinerseits $\sqrt[g]{m^g}\ (mod p)$.

Das geht, weil hier im Gegensatz zu echten RSA-Parametern der Modulus prim ist.

Wird der Angriff vor Abschluss des Handshake-Protokoll erfolgreich ausgeführt, dann ist dem Angreifer damit das Pre-Master-Secret bekannt. Damit kann er den Sitzungsschlüssel berechnen und die Finished Nachricht fälschen. Der Angriff bleibt unbemerkt. Von einem Implementator von SSL kann der Angriff recht leicht abgewehrt werden, indem er auf die Anzahl der Parameter im entsprechenden Feld der Hello Nachrichten prüft: RSA verwendet 2 Parameter, während Diffie-Hellmann 3 benötigt. In der Spezifikation sind solche Tests aber nicht beschrieben.

### 4.2 Version-Rollback-Angriff[5]

Ein Man-In-The-Middle ändert die SSLv3.0 Client-Hello-Nachricht um in eine Client-Hello-Nachricht, die dem Format der SSLv2.0 entspricht. Der Server antwortet dann mit der SSLv2.0 Server-Hello Nachricht, weil er glaubt, der Client unterstützt kein höheres Protokoll. Der Client empfängt die SSLv2.0 Nachricht des Servers und glaubt seinerseits, dass der Server kein höheres Protokoll unterstützt. Eine SSLv2.0-Sitzung mit all ihren Schwächen und Angriffspunkten beginnt.

Ermöglicht wird dieser Angriff dadurch, dass SSLv3.0 abwärtskompatibel sein muss zu SSLv2.0. Auch mit alten Clients bzw. Servern soll ja kommuniziert werden können. Deshalb wird, wie in Abbildung 3 zu sehen ist, die Version des Kommunikationsprotokolls, nämlich SSLv2.0, SSLv3.0 oder TLS[3], vor Beginn der Kommunikation ausgehandelt.

Dieser Angriff soll verhindert werden, indem die vom Client unterstützte höchste SSL-Version im PKCS-Padding der Client-Key-Exchange-Nachricht noch einmal genannt wird. Dieses kann vom Man-In-The-Middle nicht gefälscht werden, ohne RSA zu brechen oder den Schlüssel zu kennen. Spätestens wenn der Server die Key-Exchange-Nachricht des Client bekommt, wird er den Schwindel entdecken und das Alert-Protokoll aufrufen. Der Kunstgriff mit der PKCS wird verwendet, weil die Spezifikation aus Kompatibilitätsgründen im Nachhinein nicht mehr verändert werden darf.

## 5 Fazit

Insgesamt können TLS und SSL nur so sicher sein, wie die Implementierung, Benutzungsoberfläche, und die aufmerksame Benutzerin.

Die aktuelle Version SSLv3.0 ist mittlerweile recht sicher, auch durch verbesserte Implementierungen. Einen Blick verdient hier noch die Schlüsselerzeugung. Diese ist

in der SSLv3.0 Spezifikation festgelegt: die symmetrischen Schlüssel werden aus unverschlüsselt übertragenen Zufallswerten und aus einem kleinen Pre-Master-Secret mit Hilfe von Hash-Funktionen erzeugt. Wie sicher dieses Verfahren wirklich ist, wäre interessant zu untersuchen, ist aber nicht mehr Gegenstand dieser Ausarbeitung.

Die TLSv1.0 Spezifikation ist wesentlich expliziter, viele aus SSLv3.0 bekannte Schwachstellen sind dadurch eliminiert.

Generelle Probleme beim Entwurf einer SSL-Spezifikation stellen sich einmal dadurch, dass die SSL-Versionen abwärtskompatibel sein müssen: das schränkt die Möglichkeiten ein und schafft, wie gesehen, Angriffspunkte.

Veröffentlichte Spezifikationen dürfen nicht mehr verändert werden, um Inkompatibilitäten zwischen den Implementierungen zu vermeiden. Angriffspunkte, die nach der Veröffentlichung entdeckt werden, müssen daher gesichert werden ohne die Spezifikation zu ändern.

Ein weiteres Problem ist, dass SSL-Verbindungen zu Partnern hergestellt werden müssen, die vorher unbekannt waren: ein Schlüsselaustausch vorab ist selten möglich. Daher bleiben die ersten Handshake-Nachrichten zwangsläufig unverschlüsselt, was Angriffe erleichtert.

Zuletzt ist der Wunsch nach Transparenz für den Benutzer problematisch. Es gibt bisher meines Wissens keine Untersuchungen darüber, wie man Benutzungsoberflächen so gestaltet, dass sie bei guter Benutzbarkeit auch für Unerfahrene nicht die Sicherheit beeinträchtigen. [8]

# Literatur

## 5.1 SSL und TLS

[1] E. Kipp, B. Hickman. SSL 2.0 Protocol Spezification; `http://www.netscape.com/eng/security/SSL_2.html`

Die SSLv2.0 Spezifikation.

[2] Alan O. Freier, Philip Karlton, Paul C. Kocher. The SSL Protocol Version 3.0; Internet-Draft, `http://home.netscape.com/eng/ssl3/ssl-toc.html`, November 1996.

Die SSLv3.0 Spezifikation. Die Grundlage für den Vortrag.

[3] T. Dierks, C. Allen. The TLS Protocol; Request for Comments: 2246, `http://www.ietf.org/rfc/rfc2246.txt?number=2246`, 1999.

Die neueste SSL-Version. Besser zu lesen als die SSLv3.0-Spezifikation und im Kern sehr ähnlich.

## 5.2 Analyse von SSL

[4] B. Pfitzmann. Script zur Vorlesung Kryptographie, Kapitel 12, Robuster Protokollentwurf; `http://www-krypt.cs.uni-sb.de/lehre/veranstaltungen/ss2001/kryptographie/Pfit1_01KryptoUmsortiert.pdf`; Sommersemester 2001.

Robuster Protokollentwurf. Aus der Nichtbeachtung von Regeln zum robusten Protokollentwurf ergeben sich Angriffspunkte in SSL.

[8] Bespielsweise benutzerunterstützte Zertifikatsprüfung in den Browsern.

[5] David Wagner, Bruce Schneier. Analysis of the SSL 3.0 protocol, revised version; Originalversion: 2nd USENIX Workshop on Electronic Commerce, 1996, 29-40. Revised: `http://www.counterpane.com/ssl-revised.pdf`, 1997.

Eine Analyse des SSLv3.0 Protokolls auf Schwachstellen. Es werden auch einige Angriffe auf und Schwachstellen von SSLv2.0 betrachtet.

[6] Daniel Bleichenbacher. Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS#1; Crypto 1998, LNCS 1492, Springer-Verlag, Berlin 1998, 1-12.

Der Bleichenbacher-Angriff.