Bundesamt
für Sicherheit in der
Informationstechnik

Technische Richtlinie TR-02102-2

# Kryptographische Verfahren: Empfehlungen und Schlüssellängen

Teil 2 – Verwendung von Transport Layer Security (TLS)

(Version 2014-01)

Bundesamt für Sicherheit in der Informationstechnik
Postfach 20 03 63
53133 Bonn
E-Mail: TR02102@bsi.bund.de
Internet: https://www.bsi.bund.de

# Inhaltsverzeichnis



# Tabellenverzeichnis

# 1 Einleitung

Diese Technische Richtlinie gibt Empfehlungen für den Einsatz des kryptographischen Protokolls *Transport Layer Security (TLS).* Es dient der sicheren Übertragung von Informationen in Datennetzwerken, wobei insbesondere die Vertraulichkeit, die Integrität und die Authentizität der übertragenen Informationen geschützt werden können.

Die vorliegende Richtlinie enthält Empfehlungen für die zu verwendende Protokollversion und die kryptographischen Algorithmen als Konkretisierung der allgemeinen Empfehlungen in Teil 1 dieser Technischen Richtlinie.

Diese Richtlinie enthält keine Vorgaben für konkrete Anwendungen, keine Risikobewertungen sowie keine Angriffsmöglichkeiten, die sich aus Fehlern in der Implementierung des Protokolls ergeben.

**Hinweis:** Auch bei Beachtung aller Vorgaben für die Verwendung von TLS können Daten in erheblichem Umfang aus einem kryptographischen System abfließen, z. B. durch Ausnutzung von Seitenkanälen (Messung von Timing-Verhalten, Stromaufnahme, Datenraten etc.). Daher sollte der Entwickler unter Hinzuziehung von Experten auf diesem Gebiet mögliche Seitenkanäle identifizieren und entsprechende Gegenmaßnahmen umsetzen. Je nach Anwendung gilt dies auch für Fault-Attacken.

**Hinweis:** Für Definitionen kryptographischer Begriffe in diesem Dokument siehe das Glossar in [TR-02102-1].

# 2 Grundlagen

Transport Layer Security (TLS), früher bekannt als Secure Socket Layer (SSL), ermöglicht die sichere Übertragung von Informationen aus der Anwendungsschicht (z. B. HTTPS, FTPS oder IMAPS) über TCP/IP-basierte Verbindungen (insbesondere das Internet).

Bevor Daten übermittelt werden können, muss eine (gesicherte) Verbindung zwischen den zwei Verbindungspartnern (Client und Server) aufgebaut werden. Dieser Vorgang heißt *Handshake* und ist ein wichtiger Bestandteil des TLS-Protokolls. Hierbei werden zwischen Client und Server vereinbart:

1. Kryptographische Verfahren zur *Datenverschlüsselung*, *Integritätssicherung, Schlüsseleinigung* und ggf. zur (ein- oder beidseitigen) *Authentisierung.* Diese Verfahren werden durch die *Cipher-Suite* festgelegt (siehe Abschnitt 3.3).
2. Ein gemeinsames Geheimnis, das *premaster secret*. Aus diesem wird (von beiden Verbindungspartnern) das *master secret* erzeugt, aus welchem wiederum die Sitzungsschlüssel für den Integritätsschutz und die Verschlüsselung abgeleitet werden.

**Hinweis:** Das TLS-Protokoll erlaubt auch Verbindungen, die nicht oder nur einseitig authentisiert sind (Beispiel: HTTPS-Verbindungen sind üblicherweise nur serverseitig authentisiert). Daher sollten Systementwickler darauf achten, ob eine weitere Authentisierung in der Anwendungsschicht erforderlich ist (Beispiel: Authentisierung eines Homebanking-Benutzers durch Anforderung eines Passwortes). Bei Anforderung besonders kritischer Operationen sollte dabei grundsätzlich eine Au-

thentisierung durch Wissen und Besitz erfolgen, die sich unter Ausnutzung kryptographischer Mechanismen auch auf die übertragenen Daten erstrecken sollte.

# 3 Vorgaben

## 3.1 Allgemeine Hinweise

### 3.1.1 Verwendungszeiträume

Die Vorgaben und Empfehlungen in dieser Technischen Richtlinie sind jeweils mit einem maximalen Verwendungszeitraum versehen. Die Angabe der Jahreszahl bedeutet, dass das entsprechende Verfahren bis zum Ende des angegebenen Jahres eingesetzt werden kann. Ist die Jahreszahl mit einem „+“-Zeichen gekennzeichnet, so besteht die Möglichkeit einer Verlängerung des Verwendungszeitraums.

### 3.1.2 Sicherheitsniveau

Das Sicherheitsniveau für alle kryptographischen Verfahren in dieser Technischen Richtlinie richtet sich nach dem in Abschnitt 1.1 in [TR-02102-1] angegebenen Sicherheitsniveau.

### 3.1.3 Schlüssellängen bei EC-Verfahren

Die Schlüssellängen bei Verfahren, die auf elliptischen Kurven basieren, sind – im Vergleich zum Sicherheitsniveau von RSA – in dieser Technischen Richtlinie etwas größer gewählt worden, um einen Sicherheitsspielraum für die EC-Verfahren zu erreichen (vgl. Abschnitt 3.6). Für die Begründung und weitere Erläuterungen siehe Bemerkung 4 in Kapitel 3 in [TR-02102-1].

## 3.2 SSL/TLS-Versionen

Das SSL-Protokoll existiert in den Versionen 1.0, 2.0 und 3.0, wobei die Version 1.0 nicht veröffentlicht wurde. TLS 1.0 ist eine direkte Weiterentwicklung von SSL 3.0 und wird in [RFC2246] spezifiziert. Des weiteren gibt es das TLS-Protokoll in den Versionen 1.1 und 1.2, welche in [RFC4346] und [RFC5246] spezifiziert werden.

Empfehlungen für die Wahl der TLS-Version:

- Grundsätzlich wird die Verwendung von TLS 1.2 empfohlen.
- TLS 1.1 bietet ausreichende Sicherheit, aber im Vergleich zu TLS 1.2 hat es verschiedene Nachteile (siehe Abschnitt 1.2 in [RFC5246]); z. B. benutzt TLS 1.1 als Pseudorandom-Function noch die Kombination MD5/SHA-1, während in TLS 1.2 aktuelle Hashfunktionen wie bspw. SHA-256 möglich sind. In TLS 1.1 sind noch Cipher-Suites vorhanden, die auf IDEA und DES basieren, in TLS 1.2 nicht mehr.
- TLS 1.0 kann in bestehenden Anwendungen übergangsweise weiter eingesetzt werden, sofern geeignete Schutzmaßnahmen gegen Chosen-Plaintext-Angriffe (siehe [BARD] und [BEAST]) auf die CBC-Implementierung in TLS 1.0 getroffen werden. Siehe Abschnitt 3.3.2 für den erlaubten Übergangszeitraum und weitere Details dieser Übergangsregelung.

- SSL v2 ([SSLv2]) und SSL v3 ([SSLv3]) dürfen nicht mehr eingesetzt werden (siehe auch [RFC6176]).

## 3.3 Cipher-Suites

Eine Cipher-Suite spezifiziert die zu verwendenden Algorithmen für

- die Schlüsseleinigung (und ggf. Authentisierung),
- die Nutzdaten-Verschlüsselung (Stromchiffre oder Blockchiffre inkl. Betriebsmodus), und
- eine Hashfunktion für die Integritätssicherung (HMAC-Algorithmus) der Datenpakete und für die Verwendung als Pseudozufallszahlengenerator (ab TLS 1.2).

Eine vollständige Liste aller definierten Cipher-Suites mit Verweisen auf die jeweiligen Spezifikationen ist verfügbar unter [IANA].

### 3.3.1 Empfohlene Cipher-Suites

Grundsätzlich wird empfohlen, nur Cipher-Suites einzusetzen, die die Anforderungen an die Algorithmen und Schlüssellängen aus [TR-02102-1] erfüllen.

Es wird die Verwendung der folgenden Cipher-Suites empfohlen:

| | ***Schlüsseleinigung und -authentisierung*** | | ***Verschlüsselung*** | ***Betriebs-modus*** | ***Hash*** | ***Verwendung bis*** |
|---|---|---|---|---|---|---|
| TLS_ | ECDHE_ECDSA_ | WITH_ | AES_128_ | CBC_<br>GCM_ | SHA256 | 2020+ |
| | | | AES_256_ | CBC_<br>GCM_ | SHA384 | 2020+ |
| | ECDHE_RSA_ | WITH_ | AES_128_ | CBC_<br>GCM_ | SHA256 | 2020+ |
| | | | AES_256_ | CBC_<br>GCM_ | SHA384 | 2020+ |
| | DHE_DSS_[1] | WITH_ | AES_128_ | CBC_<br>GCM_ | SHA256 | 2020+ |
| | | | AES_256_ | CBC_ | SHA256 | 2020+ |
| | | | | GCM_ | SHA384 | 2020+ |
| | DHE_RSA_[1] | WITH_ | AES_128_ | CBC_<br>GCM_ | SHA256 | 2020+ |
| | | | AES_256_ | CBC_ | SHA256 | 2020+ |
| | | | | GCM_ | SHA384 | 2020+ |

*Tabelle 1: Empfohlene Cipher-Suites mit Forward Secrecy*

1 Da einige gängige Implementierungen von DH(E) in TLS zurzeit nur 1024 Bit unterstützen, sei hier auf Abschnitt 7.2.1 in [TR-02102-1] verwiesen, in welcher eine Mindestgröße von 2000 Bit für dieses Verfahren empfohlen wird.

Sofern die Verwendung von Cipher-Suites mit Forward Secrecy nicht möglich ist[2], können auch die folgenden Cipher-Suites eingesetzt werden:

| | ***Schlüsseleinigung und -authentisierung*** | | ***Verschlüsselung*** | ***Betriebs-modus*** | ***Hash*** | ***Verwendung bis*** |
|---|---|---|---|---|---|---|
| TLS_ | ECDH_ECDSA_ | WITH_ | AES_128_ | CBC_<br>GCM_ | SHA256 | 2020+ |
| | | | AES_256_ | CBC_<br>GCM_ | SHA384 | 2020+ |
| | ECDH_RSA_ | WITH_ | AES_128_ | CBC_<br>GCM_ | SHA256 | 2020+ |
| | | | AES_256_ | CBC_<br>GCM_ | SHA384 | 2020+ |
| | DH_DSS_ | WITH_ | AES_128_ | CBC_<br>GCM_ | SHA256 | 2020+ |
| | | | AES_256_ | CBC_ | SHA256 | 2020+ |
| | | | | GCM_ | SHA384 | 2020+ |
| | DH_RSA_ | WITH_ | AES_128_ | CBC_<br>GCM_ | SHA256 | 2020+ |
| | | | AES_256_ | CBC_ | SHA256 | 2020+ |
| | | | | GCM_ | SHA384 | 2020+ |

*Tabelle 2: Empfohlene Cipher-Suites ohne Forward Secrecy*

Sofern zusätzliche vorab ausgetauschte Daten in die Schlüsseleinigung einfließen sollen (*Pre-Shared Key*), bietet TLS die Verwendung entsprechender Cipher-Suites. Es wird die Verwendung von Cipher-Suites empfohlen, bei der neben dem Pre-Shared Key weitere ephemerale Schlüssel oder ausgetauschte Zufallszahlen in die Schlüsseleinigung eingehen. Die Verwendung von `TLS_PSK_*` (d. h. ohne zusätzliche ephemerale Schlüssel/Zufallszahlen) wird *nicht* empfohlen, da bei diesen Cipher-Suites die Sicherheit der Verbindung ausschließlich auf der Entropie und der Vertraulichkeit des Pre-Shared Keys beruht.

2 Forward Secrecy (auch Perfect Forward Secrecy, kurz PFS) bedeutet, dass eine Verbindung auch bei Kenntnis der Langzeit-Schlüssel der Kommunikationspartner nicht nachträglich entschlüsselt werden kann. Bei der Verwendung von TLS zum Schutz personenbezogener oder anderer sensibler Daten ist Forward Secrecy grundsätzlich notwendig.

| | *Schlüsseleinigung und -authentisierung* | | *Verschlüsselung* | *Betriebs-modus* | *Hash* | *Verwendung bis* |
|---|---|---|---|---|---|---|
| TLS_ | ECDHE_PSK_ | WITH_ | AES_128_ | CBC_ | SHA256 | 2020+ |
| | | | AES_256_ | | SHA384 | 2020+ |
| | DHE_PSK_ | WITH_ | AES_128_ | CBC_ | SHA256 | 2020+ |
| | | | | GCM_ | | 2020+ |
| | | | AES_256_ | CBC_ | SHA384 | 2020+ |
| | | | | GCM_ | | 2020+ |
| | RSA_PSK_ | WITH_ | AES_128_ | CBC_ | SHA256 | 2020+ |
| | | | | GCM_ | | 2020+ |
| | | | AES_256_ | CBC_ | SHA384 | 2020+ |
| | | | | GCM_ | | 2020+ |

*Tabelle 3: Empfohlene Cipher-Suites mit Pre-Shared Key*

Die Cipher-Suites TLS_RSA_PSK_* in Tabelle 3 bieten *keine* Forward Secrecy, alle anderen Suites aus Tabelle 3 bieten Forward Secrecy.

### 3.3.2 Übergangsregelungen

Abweichend zu obigen Vorgaben und den Empfehlungen in Teil 1 dieser Technischen Richtlinie kann in *bestehenden* Anwendungen als Hashfunktion für die Integritätssicherung mittels HMAC auch übergangsweise noch SHA-1 eingesetzt werden (d. h. Cipher-Suites der Form `*_SHA`). Unabhängig von der in Tabelle 4 angegebenen *maximalen* Verwendung wird eine schnellstmögliche Migration zu SHA-256 bzw. SHA-384 und TLS 1.2 empfohlen.

**Hinweis:** Da TLS 1.1 die Hashfunktion SHA-1 verwendet (und keine Unterstützung der SHA-2-Familie bietet), darf diese Protokoll-Version nach der Abkündigung von SHA-1 nicht mehr benutzt werden.

Der Verschlüsselungsalgorithmus RC4 in TLS weist erhebliche Sicherheitsschwächen auf und darf nicht mehr eingesetzt werden.

TLS 1.0 kann in bestehenden Anwendungen übergangsweise weiter eingesetzt werden, falls

(a) eine sofortige Migration zu TLS 1.1 oder TLS 1.2 nicht möglich ist *und*

(b) geeignete Schutzmaßnahmen gegen Chosen-Plaintext-Angriffe (siehe [BARD] und [BEAST]) auf die CBC-Implementierung in TLS 1.0 getroffen werden.

| *Abweichung* | *Verwendung maximal bis* | *Empfehlung* |
|---|---|---|
| SHA-1 als Komponente für die Signaturerstellung[3] | 2015 | Migration zu SHA-256/-384 |
| SHA-1 zur HMAC-Berechnung und als Komponente der PRF[3] | 2017+ | |
| RC4 als Verschlüsselungsfunktion | 2013 | Migration zu TLS 1.2 mit AES |
| TLS 1.0 zusammen mit Schutzmaßnahmen wie oben beschrieben | 2014 | |

*Tabelle 4: Übergangsregelungen*

## 3.4 Weitere Hinweise und Empfehlungen zu TLS

### 3.4.1 Session Renegotiation

*Session Renegotiation* darf nur auf Basis von [RFC5746] verwendet werden. Durch den Client initiierte Renegotiation sollte vom Server abgelehnt werden.

### 3.4.2 Verkürzung der HMAC-Ausgabe

Die in [RFC6066] definierte Extension „truncated_hmac“ zur Verkürzung der Ausgabe des HMAC auf 80 Bit sollte *nicht* verwendet werden.

### 3.4.3 TLS-Kompression und der CRIME-Angriff

TLS bietet die Möglichkeit, die übertragenen Daten vor der Verschlüsselung zu komprimieren. Dies führt zu der Möglichkeit eines Seitenkanalangriffes auf die Verschlüsselung über die Länge der verschlüsselten Daten (siehe [CRIME]).

Um dies zu verhindern, muss sichergestellt werden, dass alle Daten eines Datenpakets von dem korrekten und legitimen Verbindungspartner stammen und keine Plaintext-Injection durch einen Angreifer möglich ist. Kann dies nicht sichergestellt werden, so darf die TLS-Datenkompression nicht verwendet werden.

## 3.5 Authentisierung der Kommunikationspartner

Das TLS-Protokoll bietet die folgenden drei Möglichkeiten zur Authentisierung der Kommunikationspartner:

3 Aufgrund von Angriffen gegen die Kollisionsresistenz-Eigenschaften von SHA-1 (siehe auch Abschnitt 1.4 und Bemerkung 10 in [TR-02102-1]), muss darauf geachtet werden, ob die Kollisionsresistenz beim Einsatz von SHA-1 benötigt wird. Bei der Signaturerstellung ist dies der Fall, daher darf SHA-1 nur bis Ende 2015 eingesetzt werden. Bei der HMAC- und PRF-Berechnung wird die Kollisionsresistenz nicht benötigt, daher ist der maximale Einsatzzeitraum größer als bei der Signaturerstellung.

Es handelt sich hierbei um eine Ausnahmeregelung für bestehende Systeme, die der Tatsache geschuldet ist, dass erst ab TLS 1.2 mit Hilfe der `signature_algorithms`-Erweiterung die Möglichkeit besteht, dass Client und Server das Signaturverfahren und die Hashfunktion aushandeln können.

- Authentisierung beider Kommunikationspartner
- Nur serverseitige Authentisierung
- Keine Authentisierung

Die Notwendigkeit einer Authentisierung ist anwendungsabhängig. Bei der Verwendung von TLS im Web ist im Allgemeinen zumindest eine Authentisierung des Servers notwendig. Bei der Verwendung in geschlossenen Systemen (VPN o. ä.) ist zumeist eine beidseitige Authentisierung notwendig.

Für die Authentisierung bei der Verwendung in Projekten des Bundes sind die Vorgaben in [TR-03116-4] zu beachten.

## 3.6 Domainparameter und Schlüssellängen

Die Domainparameter und Schlüssellängen für

- statische Schlüsselpaare der Kommunikationspartner,
- ephemerale Schlüsselpaare bei der Verwendung von Cipher-Suites mit Forward Secrecy, und
- Schlüsselpaare für die Signatur von Zertifikaten

müssen den Vorgaben in Teil 1 dieser Technischen Richtlinie entsprechen. Es wird empfohlen, mindestens die folgenden Schlüssellängen zu verwenden:

| *Algorithmus* | *Minimale Schlüssellänge* | *Verwendung bis* |
|---|---|---|
| ***Signaturschlüssel für Zertifikate und Schlüsseleinigung*** | | |
| ECDSA | 224 Bit | 2015 |
| ECDSA | 250 Bit[4] | 2020+ |
| DSS | 2000 Bit[5] | 2020+ |
| RSA | 2000 Bit[5] | 2020+ |
| ***Statische Diffie-Hellman Schlüssel*** | | |
| ECDH | 224 Bit | 2015 |
| ECDH | 250 Bit[4] | 2020+ |
| DH | 2000 Bit[5] | 2020+ |
| ***Ephemerale Diffie-Hellman Schlüssel*** | | |
| ECDH | 224 Bit | 2015 |
| ECDH | 250 Bit[4] | 2020+ |
| DH | 2000 Bit[5] | 2020+ |

*Tabelle 5: Empfohlene Schlüssellängen*

**Hinweis:** Ist ein Schlüsselpaar *statisch,* so wird dieses mehrfach für neue Verbindungen wiederverwendet. Im Gegensatz dazu bedeutet *ephemeral,* dass für jede neue Verbindung ein neues Schlüsselpaar erzeugt wird. Ephemerale Schlüssel müssen nach Verbindungsende sicher gelöscht werden, siehe Abschnitt 4.2.

Beim Einsatz von elliptischen Kurven müssen stets kryptographisch starke Kurven über endlichen Körpern der Form $F_p$ (p prim) verwendet werden. Zusätzlich wird empfohlen, nur *named curves* (siehe [IANA]) einzusetzen, um Angriffe über nicht verifizierte schwache Domainparameter zu verhindern. Die folgenden *named curves* werden empfohlen:

- brainpoolP256r1, brainpoolP384r1, brainpoolP512r1 (vgl. [RFC5639] und [RFC7027])

Sollten diese Kurven nicht verfügbar sein, so können auch die folgenden Kurven eingesetzt werden:

- secp224r1, secp256r1, secp384r1.

# 4 Schlüssel und Zufallszahlen

## 4.1 Schlüsselspeicherung

Private kryptographische Schlüssel, insbesondere statische Schlüssel und Signaturschlüssel, müssen sicher gespeichert und verarbeitet werden. Dies bedeutet u. a. den Schutz vor Kopieren, missbräuch-

4 Hier werden 250 Bit (statt 256 Bit) festgelegt, um kleine Co-Faktoren bei elliptischen Kurven zu ermöglichen.

5 Für einen Einsatzzeitraum nach 2015 kann es sinnvoll sein, RSA/DSS/DH-Schlüssel von 3000 Bit Länge zu verwenden, um ein gleichmäßiges Sicherheitsniveau in allen empfohlenen asymmetrischen Verschlüsselungsverfahren zu erzielen.

licher Nutzung und Manipulation der Schlüssel. Eine sichere Schlüsselspeicherung kann z. B. durch die Verwendung entsprechend zertifizierter Hardware (Chipkarte, HSM) gewährleistet werden.

Ebenso müssen die öffentlichen Schlüssel von als vertrauenswürdig erkannten Stellen (Vertrauensanker) manipulationssicher gespeichert werden.

## 4.2 Umgang mit Ephemeralschlüsseln

Wenn eine Cipher-Suite mit Forward Secrecy verwendet wird, muss sichergestellt werden, dass alle Ephemeralschlüssel nach ihrer Verwendung (Ende der Verbindung) unwiderruflich gelöscht werden, und keine Kopien dieser Schlüssel erzeugt wurden. Ephemeral- bzw. Sitzungsschlüssel dürfen nur für *eine* Verbindung benutzt werden und sollten grundsätzlich nicht persistent abgespeichert werden.

## 4.3 Zufallszahlen

Für die Erzeugung von Zufallszahlen, z. B. für kryptographische Schlüssel oder die Signaturerzeugung, müssen geeignete Zufallszahlengeneratoren eingesetzt werden.

Empfohlen wird ein Zufallszahlengenerator aus einer der Klassen DRG.3, DRG.4, PTG.3 oder NTG.1 gemäß [AIS 20/31], vgl. auch Kapitel 9 in Teil 1 dieser Technischen Richtlinie.

# Literaturverzeichnis


[AIS 20/31] BSI: AIS 20/31 -- A proposal for: Functionality classes for random number generators

[TR-02102-1] BSI: Technische Richtlinie TR-02102-1, Kryptographische Verfahren: Empfehlungen und Schlüssellängen, Version 2014-01

[TR-03116-4] BSI: TR-03116 Kryptographische Vorgaben für Projekte der Bundesregierung, Teil 4 - Kommunikationsverfahren im eGovernment

[BARD] Gregory V. Bard: A challenging but feasible blockwise-adaptive chosen-plaintext attack on SSL (2006), http://eprint.iacr.org/2006/136

[IANA] IANA: http://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml

[RFC2246] IETF: T. Dierks, C. Allen: RFC 2246, The TLS Protocol Version 1.0

[RFC4346] IETF: T. Dierks, E. Rescorla: RFC 4346, The Transport Layer Security (TLS) Protocol Version 1.1

[RFC5246] IETF: T. Dierks, E. Rescorla: RFC 5246, The Transport Layer Security (TLS) Protocol Version 1.2

[RFC5639] IETF: M. Lochter, J. Merkle: RFC 5639, Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation

[RFC5746] IETF: E. Rescorla, M. Ray, S. Dispensa, N. Oskov: RFC 5746, Transport Layer Security (TLS) Renegotiation Indication Extension

[RFC6066] IETF: D. Eastlake 3rd: RFC 6066, Transport Layer Security (TLS) Extensions: Extension Definitions

[RFC6176] IETF: S. Turner, T. Polk: RFC 6176, Prohibiting Secure Sockets Layer (SSL) Version 2.0

[RFC7027] IETF: M. Lochter, J. Merkle: RFC 7027, Elliptic Curve Cryptography (ECC) Brainpool Curves for Transport Layer Security (TLS)

[BEAST] J. Rizzo, Th. Duong: BEAST: Surprising crypto attack against HTTPS, http://www.ekoparty.org/2011/juliano-rizzo.php

[CRIME] J. Rizzo, Th. Duong: The CRIME attack, http://www.ekoparty.org/2012/thai-duong.php

[SSLv2] Netscape: Hickman, Kipp: "The SSL Protocol"

[SSLv3] Netscape: A. Frier, P. Karlton, P. Kocher: "The SSL 3.0 Protocol"