

Network Working Group
Request for Comments: 2246
Category: Standards Track

T. Dierks
Certicom
C. Allen
Certicom
January 1999

The TLS Protocol
Version 1.0

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (1999). All Rights Reserved.

Abstract

This document specifies Version 1.0 of the Transport Layer Security (TLS) protocol. The TLS protocol provides communications privacy over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery.

Table of Contents

1.	Introduction	3
2.	Goals	4
3.	Goals of this document	5
4.	Presentation language	5
4.1.	Basic block size	6
4.2.	Miscellaneous	6
4.3.	Vectors	6
4.4.	Numbers	7
4.5.	Enumerateds	7
4.6.	Constructed types	8
4.6.1.	Variants	9
4.7.	Cryptographic attributes	10
4.8.	Constants	11
5.	HMAC and the pseudorandom function	11
6.	The TLS Record Protocol	13
6.1.	Connection states	14

6.2.	Record layer	16
6.2.1.	Fragmentation	16
6.2.2.	Record compression and decompression	17
6.2.3.	Record payload protection	18
6.2.3.1.	Null or standard stream cipher	19
6.2.3.2.	CBC block cipher	19
6.3.	Key calculation	21
6.3.1.	Export key generation example	22
7.	The TLS Handshake Protocol	23
7.1.	Change cipher spec protocol	24
7.2.	Alert protocol	24
7.2.1.	Closure alerts	25
7.2.2.	Error alerts	26
7.3.	Handshake Protocol overview	29
7.4.	Handshake protocol	32
7.4.1.	Hello messages	33
7.4.1.1.	Hello request	33
7.4.1.2.	Client hello	34
7.4.1.3.	Server hello	36
7.4.2.	Server certificate	37
7.4.3.	Server key exchange message	39
7.4.4.	Certificate request	41
7.4.5.	Server hello done	42
7.4.6.	Client certificate	43
7.4.7.	Client key exchange message	43
7.4.7.1.	RSA encrypted premaster secret message	44
7.4.7.2.	Client Diffie-Hellman public value	45
7.4.8.	Certificate verify	45
7.4.9.	Finished	46
8.	Cryptographic computations	47
8.1.	Computing the master secret	47
8.1.1.	RSA	48
8.1.2.	Diffie-Hellman	48
9.	Mandatory Cipher Suites	48
10.	Application data protocol	48
A.	Protocol constant values	49
A.1.	Record layer	49
A.2.	Change cipher specs message	50
A.3.	Alert messages	50
A.4.	Handshake protocol	51
A.4.1.	Hello messages	51
A.4.2.	Server authentication and key exchange messages	52
A.4.3.	Client authentication and key exchange messages	53
A.4.4.	Handshake finalization message	54
A.5.	The CipherSuite	54
A.6.	The Security Parameters	56
B.	Glossary	57
C.	CipherSuite definitions	61

D.	Implementation Notes	64
D.1.	Temporary RSA keys	64
D.2.	Random Number Generation and Seeding	64
D.3.	Certificates and authentication	65
D.4.	CipherSuites	65
E.	Backward Compatibility With SSL	66
E.1.	Version 2 client hello	67
E.2.	Avoiding man-in-the-middle version rollback	68
F.	Security analysis	69
F.1.	Handshake protocol	69
F.1.1.	Authentication and key exchange	69
F.1.1.1.	Anonymous key exchange	69
F.1.1.2.	RSA key exchange and authentication	70
F.1.1.3.	Diffie-Hellman key exchange with authentication	71
F.1.2.	Version rollback attacks	71
F.1.3.	Detecting attacks against the handshake protocol	72
F.1.4.	Resuming sessions	72
F.1.5.	MD5 and SHA	72
F.2.	Protecting application data	72
F.3.	Final notes	73
G.	Patent Statement	74
	Security Considerations	75
	References	75
	Credits	77
	Comments	78
	Full Copyright Statement	80

1. Introduction

The primary goal of the TLS Protocol is to provide privacy and data integrity between two communicating applications. The protocol is composed of two layers: the TLS Record Protocol and the TLS Handshake Protocol. At the lowest level, layered on top of some reliable transport protocol (e.g., TCP[TCP]), is the TLS Record Protocol. The TLS Record Protocol provides connection security that has two basic properties:

- The connection is private. Symmetric cryptography is used for data encryption (e.g., DES [DES], RC4 [RC4], etc.) The keys for this symmetric encryption are generated uniquely for each connection and are based on a secret negotiated by another protocol (such as the TLS Handshake Protocol). The Record Protocol can also be used without encryption.
- The connection is reliable. Message transport includes a message integrity check using a keyed MAC. Secure hash functions (e.g., SHA, MD5, etc.) are used for MAC computations. The Record Protocol can operate without a MAC, but is generally only used in

this mode while another protocol is using the Record Protocol as a transport for negotiating security parameters.

The TLS Record Protocol is used for encapsulation of various higher level protocols. One such encapsulated protocol, the TLS Handshake Protocol, allows the server and client to authenticate each other and to negotiate an encryption algorithm and cryptographic keys before the application protocol transmits or receives its first byte of data. The TLS Handshake Protocol provides connection security that has three basic properties:

- The peer's identity can be authenticated using asymmetric, or public key, cryptography (e.g., RSA [RSA], DSS [DSS], etc.). This authentication can be made optional, but is generally required for at least one of the peers.
- The negotiation of a shared secret is secure: the negotiated secret is unavailable to eavesdroppers, and for any authenticated connection the secret cannot be obtained, even by an attacker who can place himself in the middle of the connection.
- The negotiation is reliable: no attacker can modify the negotiation communication without being detected by the parties to the communication.

One advantage of TLS is that it is application protocol independent. Higher level protocols can layer on top of the TLS Protocol transparently. The TLS standard, however, does not specify how protocols add security with TLS; the decisions on how to initiate TLS handshaking and how to interpret the authentication certificates exchanged are left up to the judgment of the designers and implementors of protocols which run on top of TLS.

2. Goals

The goals of TLS Protocol, in order of their priority, are:

1. Cryptographic security: TLS should be used to establish a secure connection between two parties.
2. Interoperability: Independent programmers should be able to develop applications utilizing TLS that will then be able to successfully exchange cryptographic parameters without knowledge of one another's code.
3. Extensibility: TLS seeks to provide a framework into which new public key and bulk encryption methods can be incorporated as necessary. This will also accomplish two sub-goals: to prevent

the need to create a new protocol (and risking the introduction of possible new weaknesses) and to avoid the need to implement an entire new security library.

4. Relative efficiency: Cryptographic operations tend to be highly CPU intensive, particularly public key operations. For this reason, the TLS protocol has incorporated an optional session caching scheme to reduce the number of connections that need to be established from scratch. Additionally, care has been taken to reduce network activity.

3. Goals of this document

This document and the TLS protocol itself are based on the SSL 3.0 Protocol Specification as published by Netscape. The differences between this protocol and SSL 3.0 are not dramatic, but they are significant enough that TLS 1.0 and SSL 3.0 do not interoperate (although TLS 1.0 does incorporate a mechanism by which a TLS implementation can back down to SSL 3.0). This document is intended primarily for readers who will be implementing the protocol and those doing cryptographic analysis of it. The specification has been written with this in mind, and it is intended to reflect the needs of those two groups. For that reason, many of the algorithm-dependent data structures and rules are included in the body of the text (as opposed to in an appendix), providing easier access to them.

This document is not intended to supply any details of service definition nor interface definition, although it does cover select areas of policy as they are required for the maintenance of solid security.

4. Presentation language

This document deals with the formatting of data in an external representation. The following very basic and somewhat casually defined presentation syntax will be used. The syntax draws from several sources in its structure. Although it resembles the programming language "C" in its syntax and XDR [XDR] in both its syntax and intent, it would be risky to draw too many parallels. The purpose of this presentation language is to document TLS only, not to have general application beyond that particular goal.

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts



Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research



With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips



Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.