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., AES [AES], RC4 [SCH], 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-1, 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], DSA [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 to the judgment of the designers and implementors of protocols that run on top of TLS.

1.1. Requirements Terminology

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119 [REQ].

1.2. Major Differences from TLS 1.1

This document is a revision of the TLS 1.1 [TLS1.1] protocol which contains improved flexibility, particularly for negotiation of cryptographic algorithms. The major changes are:

- The MD5/SHA-1 combination in the pseudorandom function (PRF) has been replaced with cipher-suite-specified PRFs. All cipher suites in this document use P_SHA256.
- The MD5/SHA-1 combination in the digitally-signed element has been replaced with a single hash. Signed elements now include a field that explicitly specifies the hash algorithm used.
- Substantial cleanup to the client's and server's ability to specify which hash and signature algorithms they will accept. Note that this also relaxes some of the constraints on signature and hash algorithms from previous versions of TLS.
- Addition of support for authenticated encryption with additional data modes.
- TLS Extensions definition and AES Cipher Suites were merged in from external [TLSEXT] and [TLSAES].
- Tighter checking of EncryptedPreMasterSecret version numbers.
- Tightened up a number of requirements.
- Verify_data length now depends on the cipher suite (default is still 12).
- Cleaned up description of Bleichenbacher/Klima attack defenses.
- Alerts MUST now be sent in many cases.
- After a certificate_request, if no certificates are available, clients now MUST send an empty certificate list.
- TLS_RSA_WITH_AES_128_CBC_SHA is now the mandatory to implement cipher suite.
- Added HMAC-SHA256 cipher suites.
- Removed IDEA and DES cipher suites. They are now deprecated and will be documented in a separate document.

- Support for the SSLv2 backward-compatible hello is now a MAY, not a SHOULD, with sending it a SHOULD NOT. Support will probably become a SHOULD NOT in the future.
- Added limited "fall-through" to the presentation language to allow multiple case arms to have the same encoding.
- Added an Implementation Pitfalls sections
- The usual clarifications and editorial work.