



TLS Specification for Storage Systems

Version 2.1

ABSTRACT: This document specifies the requirements and guidance for use of the Transport Layer Security (TLS) protocol in conjunction with data storage technologies. The requirements are intended to facilitate secure interoperability of storage clients and servers as well as non-storage technologies that may have similar interoperability needs. This document was developed with the expectation that future versions of SMI-S and CDMI could leverage these requirements to ensure consistency between these standards as well as to more rapidly adjust the security functionality in these standards.

This document has been released and approved by the SNIA. The SNIA believes that the ideas, methodologies and technologies described in this document accurately represent the SNIA goals and are appropriate for widespread distribution. Suggestions for revisions should be directed to <https://www.snia.org/feedback/>.

SNIA Standard

October 8, 2022

USAGE

The SNIA hereby grants permission for individuals to use this document for personal use only, and for corporations and other business entities to use this document for internal use only (including internal copying, distribution, and display) provided that:

1. Any text, diagram, chart, table, or definition reproduced shall be reproduced in its entirety with no alteration, and,
2. Any document, printed or electronic, in which material from this document (or any portion hereof) is reproduced shall acknowledge the SNIA copyright on that material, and shall credit the SNIA for granting permission for its reuse.

Other than as explicitly provided above, you may not make any commercial use of this document, sell any or this entire document, or distribute this document to third parties. All rights not explicitly granted are expressly reserved to SNIA.

Permission to use this document for purposes other than those enumerated above may be requested by e-mailing tcmd@snia.org. Please include the identity of the requesting individual and/or company and a brief description of the purpose, nature, and scope of the requested use.

All code fragments, scripts, data tables, and sample code in this SNIA document are made available under the following license:

BSD 3-Clause Software License

Copyright (c) 2022, The Storage Networking Industry Association.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

* Neither the name of The Storage Networking Industry Association (SNIA) nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

DISCLAIMER

The information contained in this publication is subject to change without notice. The SNIA makes no warranty of any kind with regard to this specification, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The SNIA shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this specification.

Suggestions for revisions should be directed to <http://www.snia.org/feedback/>.

Copyright © 2022 SNIA. All rights reserved. All other trademarks or registered trademarks are the property of their respective owners

Revision History

Revision	Date	Sections	Originator:	Comments
Version 0.5 rev 1	March 29, 2013	All	Eric Hibbard	SNIA Public Review Draft (prepared for TC)
Version 0.6 rev 1	August 7, 2013	All	Eric Hibbard	Post SNIA Public Review Draft
Version 0.6 rev 2	August 19, 2013	All	Eric Hibbard	Further cleanup from SNIA Public Review
Version 0.6 rev 3	August 20, 2013	All	Eric Hibbard	Added draft PSK guidance
Version 1.0 rev 1	September 13, 2013	All	Eric Hibbard	Prepared final text
Version 1.0 rev 2	September 17, 2013	All	Eric Hibbard	Fixed an issue with certificate revocation
Version 1.01	November 20, 2014	All	Eric Hibbard	Replaced the definition for "security strength" and minor editorial fixes.
Version 1.1	December 17, 2019	All	Eric Hibbard	Updated references, removed TLS 1.0 and 1.1, added TLS 1.3 details, and clarified several elements.
Version 2.0	October 27, 2020	6	Eric Hibbard	Changed validity period, aligned recommended cipher suites with RFC 7525, and clarified requirements associated with ECDSA signature certificate
Version 2.1	July 19, 2022	All	Eric Hibbard	112 bits security strength to 128 bits and PSK support for TLS 1.3.

Suggestion for changes or modifications to this document should be submitted at <http://www.snia.org/feedback/>.

Contents

Page

INTRODUCTION	VI
1 SCOPE	1
2 NORMATIVE REFERENCES	1
3 TERMS AND DEFINITIONS.....	1
4 SYMBOLS AND ABBREVIATED TERMS.....	2
5 OVERVIEW AND CONCEPTS.....	4
5.1 GENERAL.....	4
5.2 STORAGE SPECIFICATIONS	4
5.3 OVERVIEW OF TLS	4
5.3.1 <i>TLS background</i>	4
5.3.2 <i>TLS functionality</i>	5
5.3.3 <i>Summary of cipher suites</i>	5
5.3.4 <i>X.509 digital certificates</i>	7
5.3.5 <i>Quantum computing and TLS</i>	7
6 REQUIREMENTS	8
6.1 TLS PROTOCOL REQUIREMENTS	8
6.2 CIPHER SUITES.....	8
6.2.1 <i>Required cipher suites for interoperability with TLS 1.2</i>	8
6.2.2 <i>Recommended cipher suites for enhanced security with TLS 1.2</i>	9
6.2.3 <i>Recommended cipher suites and extensions with TLS 1.3</i>	10
6.3 DIGITAL CERTIFICATES	11
6.3.1 <i>Certificate profile requirements</i>	11
6.3.2 <i>Certificate validity and path validation requirements</i>	11
6.3.3 <i>Certificate encoding requirements</i>	12
6.4 COMPRESSION METHODS	12
7 GUIDANCE FOR THE IMPLEMENTATION AND USE OF TLS IN DATA STORAGE.....	12
7.1 DIGITAL CERTIFICATES	12
7.1.1 <i>Certificate model</i>	12
7.1.2 <i>Chain of trust</i>	12
7.1.3 <i>Certificate lifecycle</i>	12
7.1.4 <i>Revocation</i>	13
7.2 SECURITY AWARENESS	13
7.3 CIPHER SUITES.....	13
7.4 USING TLS WITH HTTP	14
7.5 USE OF PRE-SHARED KEYS	14
BIBLIOGRAPHY	16

Introduction

Within information and communications technology, one of the best defenses against telecommunications attacks is to deploy security services implemented with mechanisms specified in standards that are thoroughly vetted in the public domain and rigorously tested by third party laboratories, by vendors, and by users of commercial off-the-shelf products. Three services that most often address network user security requirements are confidentiality, message integrity and authentication.

The Internet Engineering Task Force (IETF) with its Transport Layer Security (TLS) has a standard that is able to prevent tampering, message forgery, and eavesdropping by encrypting data units, or segments, from one end of the transport layer to the other. In addition, TLS is application protocol independent, which means higher-level protocols like the Hypertext Transfer Protocol (HTTP) can layer on top of the TLS protocol transparently.

Additional details beyond the basic TLS protocol specification are necessary to ensure both security and interoperability. This specification provides that detail in the form of specific requirements and guidance for using TLS in conjunction with storage systems.

Information technology — Security techniques — TLS specification for storage systems

1 Scope

This specification details the requirements for use of the Transport Layer Security (TLS) protocol in conjunction with data storage technologies. The requirements set out in this specification are intended to facilitate secure interoperability of storage clients and servers as well as non-storage technologies that may have similar interoperability needs.

This specification is relevant to anyone involved in owning, operating or using data storage devices. This includes senior managers, acquirers of storage product and service, and other non-technical managers or users, in addition to managers and administrators who have specific responsibilities for information security and/or storage security, storage operation, or who are responsible for an organization's overall security program and security policy development. It is also relevant to anyone involved in the planning, design and implementation of the architectural aspects of storage security.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 27000, *Information technology — Security techniques — Information security management systems — Overview and vocabulary*

IETF RFC 5280, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, IETF, May 2008

IETF RFC 5246, *The Transport Layer Security (TLS) Protocol Version 1.2*, IETF, August 2008

IETF RFC 5746, *Transport Layer Security (TLS) Renegotiation Indication Extension*, IETF, February 2010

IETF RFC 8446, *The Transport Layer Security (TLS) Protocol Version 1.3*, IETF, August 2018

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 27000 and the following apply.

3.1

cipher suite

named combination of authentication, encryption, and message authentication code algorithms used to negotiate the security settings for a network connection

Note 1 to entry: Cipher suites are typically used with the Transport Layer Security (TLS) and the Secure Sockets Layer (SSL) network protocols.

3.2 digital certificate

data structure signed with a digital signature that is based on a public key and which asserts that the key belongs to a subject identified in the structure

3.3 perfect forward secrecy

security condition in which a leaving entity cannot obtain any subsequent shared secret keys

[SOURCE: ISO/IEC 11770-5:2011, 3.24]

3.4 proxy

intermediary that acts as both a server and a client for the purpose of making requests on behalf of other clients

3.5 self-signed certificate

digital certificate (3.2) that is signed by the same entity whose identity it certifies

Note 1 to entry: A self-signed certificate is one signed with its own private key.

3.6 security strength

a number associated with the amount of work (i.e. the number of operations) that is required to break a cryptographic algorithm or system

Note 1 to entry: Security strength is specified in bits, and is a specific value from the set {80, 112, 128, 192, 256}. A security strength of b bits means that of the order of 2^b operations are required to break the system.

[SOURCE: ISO/IEC 9797-2:2011, 3.14]

4 Symbols and abbreviated terms

AEAD	Authenticated Encryption with Additional Data
AES	Advanced Encryption Standard
ASCII	American Standard Code for Information Interchange
CA	certificate authority
CBC	cipher block chaining
CDMI	Cloud Data Management Interface
CPU	central processing unit
CRL	certificate revocation list
DER	distinguished encoding rules

DHE	Ephemeral Diffie-Hellman
DNS	Domain Name Service
DSA	Digital Signature Algorithm
DTLS	Datagram Transport Layer Security
ECDHE	Elliptic Curve Ephemeral Diffie–Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
GCM	Galois/Counter Mode
HKDF	HMAC key derivation function
HMAC	Hash Message Authentication Code
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IANA	Internet Assigned Numbers Authority
IETF	Internet Engineering Task Force
IP	Internet Protocol
MAC	message authentication code
MD5	Message Digest 5
OCSP	Online Certificate Status Protocol
PEM	Privacy Enhanced Mail
PKCS	Public-Key Cryptography Standards
PKI	public key infrastructure
PRF	pseudorandom function
PSK	pre-shared key
RFC	Request For Comment
RSA	Rivest, Shamir, and Adelman algorithm
SHA	Secure Hash Algorithm
SMI-S	Storage Management Initiative – Specification
SNIA	Storage Networking Industry Association
SSL	Secure Socket Layer

TCP	Transmission Control Protocol
TLS	Transport Layer Security

5 Overview and concepts

5.1 General

Data storage systems and infrastructure increasingly use technologies such as protocols over TCP/IP to manage the systems and data as well as to access the data. In many situations, the historical reliance on isolated connectivity, specialized technologies, and the physical security of data centers are not sufficient to protect data, especially when the data is considered sensitive and/or high value. Thus, there is a need to include security at the transport layer and at the same time, ensure interoperability.

TLS and its predecessor, the Security Socket Layer (SSL), have been used successfully to protect a wide range of communications over TCP/IP. Recognizing this fact, the storage industry has mandated the use of TLS/SSL in conjunction with the HTTP for multiple specifications (see 5.2). These storage specifications tend to be lengthy and complex, resulting in long development cycles that don't allow for rapid requirements changes due to security vulnerabilities or new attacks.

The objectives for this specification are to:

- Specify the TLS elements necessary to secure storage management and data access
- Facilitate timely updates and enhancements to the security for the storage specifications
- Ensure storage clients and systems can interoperate securely
- Support non-storage technologies that may have similar TLS interoperability needs

While many elements of this document can be relevant to Datagram Transport Layer Security (DTLS), no provisions have been added to address DTLS conformance.

5.2 Storage specifications

As a starting point, the original TLS requirements described herein were extracted from the following specification:

- ISO/IEC 17826:2012, Information technology — Cloud Data Management Interface (CDMI)
- Storage Networking Industry Association (SNIA), Storage Management Initiative – Specification (SMI-S), Version 1.6.1

These original requirements were then harmonized, eliminating minor differences.

5.3 Overview of TLS

5.3.1 TLS background

TLS is a protocol that provides communications security over networks. It allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery. TLS

is layered on top of some reliable transport protocol (e.g., TCP), and it is used for encapsulation of various higher-level protocols (e.g., HTTP).

Version 1.2 of TLS is specified in IETF RFC 5246. The more recent version 1.3 of TLS is specified in IETF RFC 8446. Earlier, and less secure, versions of TLS are also specified and in use; TLS versions 1.0 is specified in IETF RFC 2246 and TLS versions 1.1 is specified in IETF RFC 4346. The predecessor to TLS, The Secure Sockets Layer (SSL), and in particular, version 3.0 is also in use, but also considered less secure; SSL 3.0 is documented in the historical IETF RFC 6101, *The Secure Sockets Layer (SSL) Protocol Version 3.0*.

5.3.2 TLS functionality

TLS provides endpoint authentication and communications privacy over the network using cryptography. Typically, only the server is authenticated (i.e., its identity is ensured) while the client remains unauthenticated; this means that the end user (whether an individual or an application) has a measure of assurance with whom they are communicating. Mutual authentication (the identities of both endpoints are verified) requires, with few exceptions, the deployment of digital certificates on the client.

TLS involves three basic phases:

- Peer negotiation for algorithm support
- Key exchange and authentication
- Symmetric cipher encryption and message authentication

During the first phase, the client and server negotiate cipher suites (see 5.3.3), which determine the ciphers to be used, the key exchange, authentication algorithms, and the Message Authentication Codes (MACs). The key exchange and authentication algorithms are typically public key algorithms. The MACs are made up from a keyed-Hash Message Authentication Code (HMAC).

5.3.3 Summary of cipher suites

TLS cipher suite names consist of a set of mnemonics separated by underscores (i.e., “_”). A registered¹ 16-bit (4 hexadecimal digit) number, called the cipher suite index, is assigned for each defined cipher suite. The naming convention in TLS 1.3 differs from the convention shared in TLS 1.0, 1.1, and 1.2. In all TLS cipher suites, the first mnemonic is the protocol name (i.e., “TLS”). Cipher suite names in TLS 1.0, 1.1, and 1.2 have the following form:

TLS_KeyExchangeAlg_WITH_EncryptionAlg_MessageAuthenticationAlg

where:

KeyExchangeAlg consists of one (e.g., RSA, PSK, etc.) or two (e.g., ECDHE_ECDSA) mnemonics.

EncryptionAlg indicates the symmetric encryption algorithm and associated mode of operations.

MessageAuthenticationAlg is generally the hashing algorithm to be used for HMAC, if applicable.

¹ The Internet Assigned Numbers Authority (IANA) documents registries and important TLS parameters, which can be found at: <https://www.iana.org/assignments/tls-parameters/tls-parameters.xml>.

The following examples illustrate how to interpret the cipher suite names:

- `TLS_DHE_RSA_WITH_AES_256_CBC_SHA256`: Ephemeral DH is used for the key exchange. The server's ephemeral public key is authenticated using the server's RSA public key. Once the handshake is completed, the messages are encrypted using AES-256 in CBC mode. SHA-256 is used for both the PRF and HMAC computations.
- `TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`: Ephemeral ECDH is used for the key exchange. The server's ephemeral public key is authenticated using the server's ECDSA public key. Once the handshake is completed, the messages are encrypted and authenticated using AES-256 in GCM mode, and SHA-384 is used for the PRF. Since an authenticated encryption mode is used, messages neither have nor require an HMAC message authentication code.

TLS 1.3 cipher suites have the following form:

`TLS_AEAD_HASH`

where:

AEAD indicates the AEAD algorithm that is used for confidentiality, integrity, and message authentication.

HASH indicates the hashing algorithm that is used with the HKDF during key derivation.

The following examples illustrate how to interpret TLS 1.3 cipher suite names:

- `TLS_AES_256_GCM_SHA384`: Messages are encrypted and authenticated with AES-256 in GCM mode, and SHA-384 is used with the HKDF.
- `TLS_AES_128_CCM_SHA256`: Messages are encrypted and authenticated with AES-128 in CCM mode, and SHA-256 is used with the HKDF.

The negotiation of the key exchange method is handled elsewhere in the TLS 1.3 handshake.

To ensure a measure of interoperability between clients and servers, each version of TLS specifies a mandatory cipher suite² that all compliant applications are required to implement. The following is the mandatory cipher suite associated with TLS 1.2:

- `TLS_RSA_WITH_AES_128_CBC_SHA` {0x00, 0x2F}

The client always initiates the TLS session and starts cipher suite negotiation by transmitting a handshake message that lists the cipher suites (by index value) that it will accept. The server responds with a handshake message indicating which cipher suite it selected from the list or an "abort." Although the client orders its list of cipher suite by preference, starting with the most preferred, the server may choose any of the cipher suites proposed by the client. Therefore, there is no guarantee that the negotiation will select the strongest suite. If no cipher suites are mutually supported, the connection is aborted.

² Section 9 - *Mandatory Cipher Suites* of each of the corresponding TLS IETF RFCs is where these mandatory cipher suites are specified. The mandatory cipher suite is required to be implemented and supported in the absence of an application profile.

NOTE When the negotiated options, including optional public key certificates and random data for developing keying material to be used by the cryptographic algorithms, are complete, messages are exchanged to place the communications channel in a secure mode.

5.3.4 X.509 digital certificates

TLS uses X.509 version 3 public key certificates that are conformant with the *Certificate and Certificate Extension Profile* defined in Section 4 of IETF RFC 5280. This certificate and certificate revocation list (CRL) profile specifies the mandatory fields included in the certificate as well as optional fields and extensions that may be included in the certificate. These X.509 certificates use a digital signature to bind together a public key with an identity. These signatures will often be issued by a certificate authority (CA) associated with an internal or external public key infrastructure (PKI); however, an alternate approach uses self-signed certificates (the certificate is digitally signed by the very same key-pair whose public part appears in the certificate data). The trust models associated with these two approaches are very different.

NOTE Self-signed certificates can be used to form a web of trust (trust decisions are in the hands of individual users/administrators) that is considered less secure as there is no central authority for trust (e.g., no identity assurance or revocation). This reduction in overall security, which may still offer adequate protections for some environments, is accompanied by an easing of the overall complexity of implementation.

Section 6 of IETF RFC 5280 identifies the need for clients and servers to perform basic path validation, extension path validation, and CRL validation. These validations include, but are not limited to, the following:

- The certificate is a validly constructed certificate
- The signature is correct for the certificate
- The date of its use is within the validity period (i.e., it has not expired)
- The certificate has not been revoked (applies only to PKI certificates)
- The certificate chain is validly constructed considering the peer certificate plus valid issuer certificates up to the maximum allowed chain depth (applies only to PKI certificates)

X.509 digital certificates come in various formats that involve either binary or textual (ASCII) encoding. These encoded certificates can be stored in file types that have different structures that can include server certificate, the intermediate certificate, the private key (e.g., in a PKCS#12 package/file), etc. as well as possibly including a passphrase to protect a private key. It is usually possible to convert between one certificate file type to another, so support for all files types is typically not needed.

5.3.5 Quantum computing and TLS

Quantum computing is expected to have significant impacts on the security protections offered by commonly used cryptographic algorithms. In the “post-quantum” era (i.e. when real, working, large-scale quantum computers can be built), the following asymmetric encryption algorithms are expected to be vulnerable to quantum computing:

- Rivest, Shamir, and Adelman (RSA) algorithm
- Elliptic Curve Digital Signature Algorithm (ECDSA)
- Elliptic Curve Diffie-Hellman (ECDH)
- Diffie-Hellman (DH)

While the general guidance for symmetric encryption algorithms (e.g. AES) is to double the key size, this is not always an option.

The IETF RFCs associated with TLS make no provisions for quantum computing at this time. As such, this specification does not address the potential requirements for post quantum cryptography.

6 Requirements

6.1 TLS protocol requirements

Storage systems functioning as servers shall implement the TLS protocol; however, its use by clients is optional. TLS version 1.2 (specified in IETF RFC 5246) shall be implemented, and TLS version 1.3 should be implemented. Servers shall not support SSL (i.e., versions 1.0, 2.0, and 3.0) or TLS protocol versions prior to 1.2.

NOTE Implementation of TLS version 1.3 is already mandatory, disallowing all earlier versions, for one or more storage specifications. In addition, the IETF is encouraging all new TLS enabled protocols to be developed to only support TLS version 1.3.

Storage systems shall guard against renegotiation attacks (as outlined in IETF RFC 5746) for versions of TLS prior to version 1.3, using one of the following approaches:

- Option 1: Disable renegotiations³
- Option 2: Implement the TLS Renegotiation Indication Extension specified in IETF RFC 5746

Storage systems should guard against replay attacks on zero round-trip time (0-RTT) for TLS version 1.3 by disabling the 0-RTT functionality if session restarts are not required.

6.2 Cipher suites

6.2.1 Required cipher suites for interoperability with TLS 1.2

Storage systems shall not use MD5 as the default HMAC, which is different than what is specified in the cipher suite. In addition, storage systems shall support:

- selection and use of signature/hash algorithm pairs, using the *supported_signature_algorithms* mechanism in TLS 1.2
- use of SHA-256 or greater strength hashes

Storage systems shall use cipher suites that have at least 128 bits of security strength; IETF RFC 7525 suggests that implementations not negotiate cipher suites that use algorithms offering less than 128 bits of security strength. In addition, the following cipher suites shall be supported by storage systems and clients accessing them:

- TLS_RSA_WITH_AES_128_CBC_SHA {0x00, 0x2F}⁴

³ This approach may also prevent the use of client-side certificates in certain scenarios.

⁴ In the absence of an application profile standard this is the mandatory cipher suite for TLS v1.2.

— TLS_RSA_WITH_AES_128_CBC_SHA256 {0x00, 0x3C}

NOTE The use of CBC mode encryption carries the theoretical risks associated with padding oracle attacks⁵. Other encryption modes like GCM do not carry these risks.

For some environments, the use of RSA or SHA-1 may be prohibited (see NIST SP 800-52 Revision 2). While the cipher suites listed above are required to be implemented, their use may not be allowed, so consideration should be given to supporting cipher suites listed in 6.2.2.

6.2.2 Recommended cipher suites for enhanced security with TLS 1.2

Clause 6.2.1 requires the use of algorithms that have at least 128 bits of security strength with a preference for at least 128 bits of security strength. A significant number of cipher suites are acceptable, based on these criteria. However, there are additional considerations and a need to facilitate interoperability between clients and servers, so the recommendations in this subclause should be implemented.

Forward secrecy (sometimes called "perfect forward secrecy") prevents the recovery of information that was encrypted with older session keys, thus limiting the amount of time during which attacks can be successful. Many TLS cipher suites were defined that do not feature forward secrecy (e.g., TLS_RSA_WITH_AES_128_CBC_SHA256). IETF RFC 5116 defines the authenticated encryption with additional data (AEAD) algorithms that can be used to achieve forward secrecy.

In alignment with the IETF RFC 7525 recommendations of on forward secrecy, the following cipher suites should be supported by storage systems and clients accessing them:

— TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 {0x00,0x9E}
— TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 {0x00,0x9F}
— TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 {0xC0,0x2B}
— TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 {0xC0,0x2C}
— TLS_ECDHE_EDDSA_WITH_AES_128_GCM_SHA256 { 0xCC, 0xB1 }
— TLS_ECDHE_EDDSA_WITH_AES_256_GCM_SHA384 { 0xCC, 0xB2 }

Since the use of digital certificates can add complexity, especially for isolated networks that prevent certificate path verification or for mutual authentication, an alternate approach using pre-shared keys (PSK) is permitted. As such, the following PSK cipher suites should be supported by storage systems and clients accessing them:

— TLS_DHE_PSK_WITH_AES_128_GCM_SHA256 {0x00,0xAA}⁶
— TLS_PSK_WITH_AES_128_GCM_SHA256 {0x00,0xA8}
— TLS_PSK_WITH_AES_256_GCM_SHA384 {0x00,0xA9}

⁵ Additional information on padding oracle attacks can be found in the *Practical Padding Oracle Attacks* paper by Juliano Rizzo and Thai Duong.

⁶ This cipher suite is specified in the *TCG Storage Opal SSC Feature Set: PSK Secure Messaging*, Specification Version 1.00, August 2015.

It is possible that compliance with specialized requirements (e.g., NIST SP 800-52 Revision 2) or specific TLS profiles (e.g., IETF RFC 6460) can further restrict or expand the recommended cipher suites an implementation needs to implement.

6.2.3 Recommended cipher suites and extensions with TLS 1.3

If TLS 1.3 is supported, the following IETF RFC 8446 mandatory cipher suite shall be implemented:

— TLS_AES_128_GCM_SHA256 {0x13, 0x01}

The following TLS 1.3 cipher suites, recommended by IETF RFC 8446, should also be supported by storage systems and clients accessing them:

— TLS_AES_256_GCM_SHA384 {0x13, 0x02}

— TLS_CHACHA20_POLY1305_SHA256 {0x13, 0x03}

NOTE: The U.S. Government (i.e., NIST SP 800-52 Revision 2, Section 1.3) has indicated that Agencies shall support TLS 1.3 by January 1, 2024. After this date, servers shall support TLS 1.3 for both government-only and citizen or business-facing applications. In general, servers that support TLS 1.3 should be configured to use TLS 1.2 as well. However, TLS 1.2 may be disabled on servers that support TLS 1.3 if it has been determined that TLS 1.2 is not needed for interoperability.

NOTE TLS 1.3 cipher suites cannot be used with earlier versions of TLS. TLS 1.3 cannot use cipher suites from earlier versions of TLS.

Per IETF RFC 8446, TLS 1.3 compliant application shall support digital signatures with `rsa_pkcs1_sha256` (for certificates), `rsa_pss_rsae_sha256` (for CertificateVerify and certificates), and `ecdsa_secp256r1_sha256`. In addition, clients/servers shall support key exchange with `secp256r1` (NIST P-256) and should support key exchange with X25519 (default for popular open source implementations of TLS).

Clients/servers shall implement and use the following TLS extensions:

- The Supported Versions ("supported_versions")
- Cookie ("cookie")
- Signature Algorithms ("signature_algorithms")
- Signature Algorithms Certificate ("signature_algorithms_cert")
- Negotiated Groups ("supported_groups")
- Key Share ("key_share")
- Server Name Indication ("server_name")
- Pre-shared Key Support ("pre_shared_key")

6.3 Digital certificates

6.3.1 Certificate profile requirements

Storage systems (TLS servers) shall be capable of supporting one or more public key certificates and the associated private keys. The use of public key certificates is optional, but they should be used for management operations and data access.

When public key certificates are used by storage systems and clients that access these systems, the supported certificates shall be X.509 version 3 public key certificates that are conformant with the *Certificate and Certificate Extension Profile* defined in Section 4 of IETF RFC 5280. Both the public key contained in the certificate and the signature shall provide at least 128 bits of security strength.

At a minimum, TLS servers conforming to this specification shall be configured with an RSA signature certificate or an ECDSA signature certificate. For RSA server X.509 certificates, key sizes of 3072 bits or greater (evenly divisible by 8) shall be used. If the server is configured with an ECDSA signature certificate, either curve, NIST P-384 with SHA-384, or NIST P-521 with SHA-512 shall be used for the public key in the certificate; for interoperability, clients and servers should support the curve NIST P-384

6.3.2 Certificate validity and path validation requirements

The certificate validity period is the time interval during which the CA warrants that it will maintain information about the status of the certificate. Certificates used for storage systems shall not have a validity period greater than 398 days.⁷

TLS clients/servers shall validate certificates in accordance with certificate path validation rules specified in Section 6 of IETF RFC 5280. The revocation status of each certificate in the certification path (certificate chain) shall be checked.

Revocation information shall be obtained by the TLS client/server from one or more of the following locations:

- 1 CRL or Online Certificate Status Protocol (OCSP), as described IETF RFC 6960, response in the server's local store;
- 2 OCSP response from a locally-configured OCSP responder;
- 3 OCSP response from the OCSP responder location identified in the OCSP field in the Authority Information Access extension in the certificate; or
- 4 CRL from the CRL distribution points extension in the certificate.

When the local store does not have the current or a cogent⁸ CRL or OCSP response and the OCSP responder and the CRL distribution point are unavailable or inaccessible at the time of TLS session establishment, either deny the connection or accept a potentially revoked or compromised certificate. The decision to accept or reject a certificate in this situation should be made according to organizational policy.

⁷ Based on the CA/Browser Forum, *Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates*, Version 1.7.2

⁸ A CRL is considered "cogent" when the "CRL Scope" (see IETF RFC 5280) is appropriate for the certificate in question.

6.3.3 Certificate encoding requirements

Several security-related standards used on the Internet define ASN.1 data formats that are normally encoded using the Basic Encoding Rules (BER) or Distinguished Encoding Rules (DER)⁹, which are binary, octet-oriented encodings. Alternatively, the binary encoding can be substituted with a Base64 (ASCII) encoding; a common example is Privacy Enhanced Mail (PEM) file that are just Base64 encoded DER files.

Storage clients/servers shall support both binary DER and Base64 (ASCII) DER encodings. This support shall include the ability to input the corresponding file types (e.g., “.der” and “.cer” for binary DER as well as “.pem” and “.crt” for Base64 DER). In addition, clients/servers shall support input of PKSC#12 files (e.g., “.pfx” and “.p12”).

6.4 Compression methods

The use of compression may enable attackers to perform attacks using compression-based side channels. To defend against these attacks, the null compression method shall be enabled, and all other compression methods shall be disabled.

7 Guidance for the implementation and use of TLS in data storage

7.1 Digital certificates

7.1.1 Certificate model

Digital certificates are used to identify servers (or less commonly clients) and provide cryptographic keys for use in communication. These certificates can either be public key certificates or self-signed certificates (as noted in 5.3.4). Public key certificates typically provide more reliable identity assurances but require prior planning and supporting infrastructure (e.g., certificate authorities). Self-signed certificates are very easy to deploy but do not provide a reliable identity assurance, so they should not be used on anything but test systems. Organizations should make a conscious decision between the two models based on their risk profile and available resources.

7.1.2 Chain of trust

The confidence in the identity assurance provided by a certificate really depends on confidence in the entity (i.e., CA) issuing the certificate. Often a trusted CA will issue a certificate to an organization, which will then issue their own certificates (this creates the “chain of trust”). When using public key certificates, organizations should explicitly identify the certificate authorities that are allowed to issue certificates for use within the organization. These trusted CAs should be configured in the client (e.g., web browsers).

7.1.3 Certificate lifecycle

Certificates need to be issued, installed, replaced and ultimately removed/revoked. Effective governance of this certificate lifecycle is dependent on the organization developing sound policies and procedures. A commonly overlooked decision is the lifetime of the certificate. Certificates have expiration dates to specify the maximum time a certificate is valid (much like other forms of identity assurances) and at the end of their lifetime, they have to be replaced to avoid “certificate expired” errors. When setting the lifetime, carefully consider the risk, complexity of the certificate request and installation, and the number of certificates involved. For example, if the certificate/request installation process requires 1.5 person hours and there are 10,000 certificates in the organization, setting the certificate lifetime to 1 year would require 8 full-time

⁹ See ITU-T X.690.

employees (10,000 x 1.5 divided by 8 hours per workday divided by 230 workdays per year) just to replace the certificates.

Removal of certificates from devices being repurposed or leaving organization control is an essential measure to protect the organization against unauthorized access.

If a certificate is compromised (e.g., its private key is revealed to an unauthorized person) or if the certificate is no longer needed, the certificate should be revoked (see 7.1.4) to prevent its further use.

7.1.4 Revocation

Certificates need to be invalidated (revoked) when they are no longer useful or they have been compromised (e.g., the private key associated with the certificate has come into the possession of an unauthorized party). Certificate revocation is simply the process of adding a certificate to the CRL. As described in 6.3.2, clients and other certificate users are required to check the validity of a certificate (i.e., in part to verify that it has not been revoked) before making use of a certificate.

The following important certificate revocation issues should be addressed:

- The sources of the revocation information should be relevant and trustworthy
- The revocation information should be as "fresh" as possible, especially when high value or sensitive data is involved (e.g., a CRL can have lengthy expiration dates that masks the need to retrieve a more current version of the CRL)
- CRLs, similar to certificate chains, can be large so adequate provisions should be made to store and process CRLs when they are used for validation
- When using OCSP directly, users should understand that there can be privacy issues, which necessitate additional protections (e.g., using TLS with OCSP)

It is important to note that self-signed certificates cannot be revoked using the methods listed above. Revocation of a self-signed certificate is accomplished by removing it from the whitelist of trusted certificates (essentially the same as revoking trust in a CA).

7.2 Security awareness

Users training (e.g., during security awareness training) is essential in working with certificates. For example, when the organization uses public key certificates, users should be trained to never visit a site that generates a "certificate warning" (issued by an un-trusted CA, expired, etc.) and to report those warnings. It should be noted that self-signed certificates will typically generate a warning in a client web browser and users can ignore this warning if they have other reasons for being confident in the identity of the site (e.g., establishing a HTTPS session with a storage device by the network address of its management port).

7.3 Cipher suites

As described in 5.3.3, cipher suites are a core element of TLS and they determine much of the cryptography used in a TLS session. Many different cipher suites are supported within TLS and some are stronger than others. To avoid diminishing the security within a TLS session, leverage the following cipher suite related guidance:

- Both clients and servers should be configured to require use of the cipher suites identified in 6.2.

- Weaker cipher suites (e.g., all SSL v2.0 cipher suites, that use NULL encryption, that use MD5 for hashing, that use ECB modes of operation, etc.) should be explicitly excluded from clients and servers
- Clients should be configurable to only list cipher suites that meet minimum levels of security strength (e.g., minimum of 128 bits)
- Servers should be configurable to only accept cipher suites that meet minimum levels of security strength, and to choose the strongest acceptable cipher suite presented by the client
- If the client cannot negotiate use of a recommended cipher suite with the server, then the connection should fail.

As noted in clause 6.3, for certificates using RSA, the key size shall be at least 3072 bits.

7.4 Using TLS with HTTP

A serious risk exists that an adversary might be able to set up a false server or insert an unauthorized proxy in the communications path in order to capture sensitive information such as authentication credentials. The most effective countermeasure for this attack is the controlled use of server certificates with TLS, matched by client controls on certificate acceptance on the assumption that the false server will be unable to obtain an acceptable certificate. Specifically, this could be accomplished by configuring clients to always use TLS underneath HTTP authentication.

Servers may authenticate to clients through use of server certificates issued by a specifically authorized CA (or self-signed certificates with other identify assurances such as a known network address) and matching client controls specifying acceptable certificates. Alternatively, servers may be authenticated using pre-shared keys (as described in 7.5).

7.5 Use of pre-shared keys

Authentication via a PSK implements what is generally known as “source authentication” – authentication is based on the source of a communication using the correct key. This is a much relaxed method of authentication when compared with the potentially more robust authentication offered by public-key or self-signed certificates. As noted in IETF RFC 4279, PSK can be most useful in:

- Avoidance of certificate operations – symmetric cryptographic operations are much less CPU-intensive and thus appropriate for CPU-constrained environments
- Simplified key management – for some environments, it may be much more convenient to use a PSK rather than investing in the infrastructure and processes necessary to support certificates.

However, the simplicity achieved in avoiding the use of certificates does come at a price. Consider the case where the key belonging to a server has been compromised: if it's the private key from a PKI certificate, all that is required is to revoke the current certificate and issue a new one. If the new certificate is issued by a trusted CA (or the client clicks through the certificate warnings), the client can continue to connect to the server without complication. However, if the server was using PSK to authenticate to clients, it will be necessary to generate a new key for the server and then modify the configuration of each client to use the new key when authenticating the server.

PSK also makes certain kinds of attacks somewhat easier. For example, the certificate environment makes it somewhat difficult to spoof the identify of an entity as such spoofing requires knowledge of the private key associated with a given certificate. This has motivated adversarial use of rogue certificate authorities, middleperson attacks, etc., to work around this difficulty. In the PSK environment, if the adversary can gain possession of the pre-shared key, they may impersonate the server at will as authentication is provided

solely by possession of the key. This necessity of protecting the PSK from unauthorized access or disclosure heightens the criticality of sound key management practices.

When considering use of PSK, an organization should:

- Understand the numbers of keys required and the scope of those keys – for example, when authenticating a server to clients, it may be acceptable to configure the same key on all clients. With this large scope for that key, re-keying (due to cryptoperiod management or key compromise) will require reconfiguring each client. If mutual authentication is necessary, a unique pair of keys (one at the server and one at the client) will be required for each client. Assure that the key generation process uses sufficient entropy and makes use of the entire keyspace.
- Plan the key distribution process – with PSK, keys need to be distributed between both clients and servers and it will be necessary to carefully protect those keys from disclosure during distribution.
- Protect keys stored on clients – with PSK, clients need to store the key(s) used in authenticating servers (or if mutual authentication is used, themselves to servers) and clients may maintain a more relaxed default security posture than a server in a well-managed data center. When using PSK, key storage may need to become much more robust.
- Understand the risks associated with reliance on source authentication – when authentication is achieved by knowledge of key alone, additional compensating controls may be required. For example, consider the case where a server's PSK has been compromised: an adversary can set up a rogue imitation of that server and direct clients to it (perhaps by a DNS-based exploit) and be indistinguishable from the real server. To compensate for the weakness of source authentication, it might be necessary in a particular environment to carefully segregate the network environment and employ strict access controls to limit an adversary's capability to create a rogue server within the communications scope of the clients.

Bibliography

- [01] ISO/IEC 9797-2:2011, *Information technology — Security techniques — Message Authentication Codes (MACs) — Part 2: Mechanisms using a dedicated hash-function*
- [02] ISO/IEC 11770-5:2011, *Information technology — Security techniques — Key management — Part 5: Group key management*
- [03] ISO/IEC 17826:2012, *Information technology — Cloud Data Management Interface (CDMI)*
- [04] IETF RFC 2246, *The TLS Protocol Version 1.0*, January 1999
- [05] IETF RFC 4279, *Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)*, December 2005
- [06] IETF RFC 4346, *The Transport Layer Security (TLS) Protocol Version 1.1*, April 2006
- [07] IETF RFC 5116, *An Interface and Algorithms for Authenticated Encryption*, January 2008
- [08] IETF RFC 6101, *The Secure Sockets Layer (SSL) Protocol Version 3.0*, August 2011
- [09] IETF RFC 6460, *Suite B Profile for Transport Layer Security (TLS)*, IETF, January 2012
- [10] IETF RFC 6960, *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*, June 2013
- [11] IETF RFC 7525, *Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*, May 2015
- [12] ITU-T Recommendation X.690, ISO/IEC 8825-1:2008, *Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*, November 2008
- [13] PKCS#12 Personal Information Exchange Syntax Standard
- [14] Storage Networking Industry Association (SNIA), *Storage Management Initiative – Specification (SMI-S), Version 1.6.1, Architecture Book*, http://www.snia.org/tech_activities/standards/curr_standards/smi
- [21] *Practical Padding Oracle Attacks*, Juliano Rizzo, Thai Duong, May 25, 2010, USENIX WOOT 2010, https://www.usenix.org/legacy/event/woot10/tech/full_papers/Rizzo.pdf
- [15] Trusted Computing Group (TCG) *Storage Opal SSC Feature Set: PSK Secure Messaging*, Specification Version 1.00, August 2015
- [16] NIST Special Publication 800-52 Revision 2, *Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations*, August 2019
- [17] CA/Browser Forum, *Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates*, Version 1.7.2, September 2020