Network Working Group
Internet-Draft
Intended status: Informational
Expires: 4 September 2025

A. Becker
National Security Agency

3 March 2025

Commercial National Security Algorithm (CNSA) Suite Profile for TLS 1.3
draft-becker-cnsa2-tls-profile-01

Abstract

This document defines a base profile of TLS 1.3 for use with the U.S. Commercial National Security Algorithm (CNSA) 2.0 Suite, a cybersecurity advisory published by the United States Government which outlines quantum-resistant cryptographic algorithm policy for U.S. national security applications.

This profile applies to the capabilities, configuration, and operation of all components of U.S. National Security Systems that employ TLS 1.3. It is also appropriate for all other U.S. Government systems that process high-value information.

This profile is made publicly available for use by developers and operators of these and any other system deployments.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 4 September 2025.

Copyright Notice

Table of Contents

1. Introduction

This document specifies a profile of TLS 1.3 [RFC8446] to comply with the National Security Agency's (NSA) Commercial National Security Algorithm (CNSA) 2.0 Suite [annccnsa]. This profile applies to the capabilities, configuration, and operation of all components of U.S. National Security Systems (NSS) that employ TLS 1.3. U.S. National Security Systems are described in NIST Special Publication 800-59 [SP80059]. This profile is also appropriate for all other U.S. Government systems that process high-value information, and is made

publicly available for use by developers and operators of these and any other system deployments.

This document does not define any cryptographic algorithm, and does not specify how to use any cryptographic algorithm not currently supported by TLS 1.3; instead, it profiles CNSA 2.0-compliant conventions for TLS 1.3, and uses algorithms already specified for use in TLS 1.3 that are also allowed by the CNSA Suite 2.0. This document applies to all CNSA Suite solutions that make use of TLS 1.3.

The reader is assumed to have familiarity with [RFC8446], [I-D.connolly-tls-mlkem-key-agreement], [I-D.ietf-tls-8773bis].

[ED NOTE: This document uses guidance from [I-D.connolly-tls-mlkem-key-agreement] to specify use of ML-KEM in TLS 1.3. We note that the draft is not yet an RFC, and we may need to adjust this text accordingly based on the progress of that document.]

[ED NOTE: This document will likely change to use guidance from work specifying the use of ML-DSA in TLS 1.3.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14[RFC2119] [RFC8174] if and only if they appear in all capitals, as shown here.

All MUST-level requirements of [RFC8446], [I-D.connolly-tls-mlkem-key-agreement], [I-D.ietf-tls-8773bis] apply throughout this profile; they are generally not repeated. In cases where a MUST-level requirement is repeated for clarity, the source document prevails. This profile contains changes that elevate some SHOULD-level options to MUST-level; this profile also contains changes that elevate some MAY-level options to SHOULD-level or MUST-level. All options from [RFC8446], [I-D.connolly-tls-mlkem-key-agreement], [I-D.ietf-tls-8773bis] that are not mentioned in this profile remain at the requirement level of the reference.

All references to "CNSA" in this document refer to CNSA 2.0 [annccnsa], unless stated otherwise.

3. The Commercial National Security Algorithm Suite

The National Security Agency (NSA) profiles commercial cryptographic algorithms and protocols as part of its mission to support secure, interoperable communications for U.S. Government National Security Systems (NSS). To this end, it publishes guidance both to assist with the U.S. Government transition to new algorithms and to provide vendors - and the internet community in general - with information concerning their proper use and configuration within the scope of U.S. Government National Security Systems.

The Commercial National Security Algorithm (CNSA) Suite is the set of approved commercial algorithms that can be used by vendors and IT users to meet cybersecurity and interoperability requirements for NSS. The initial suite of CNSA Suite algorithms, Suite B, established a baseline for use of commercial algorithms to protect classified information. The following suite, CNSA 1.0, served as a bridge between the original set and a fully quantum-resistant cryptographic capability. The current suite, CNSA 2.0, establishes fully quantum-resistant protection [annccnsa].

Pursuant to the National Security Memorandum on Promoting United States Leadership in Quantum Computing While Mitigating Risks to Vulnerable Cryptographic Systems [NSM-10], the National Institute for Standards and Technology (NIST) selected several quantum-resistant, or post-quantum, asymmetric algorithms for standardization. From these, NSA has included two in CNSA 2.0: ML-DSA-87 [FIPS204] for signing, and ML-KEM-1024 [FIPS203] for key establishment. ML-DSA-87 and ML-KEM-1024 together with SHA384, SHA512, AES-256, LMS, and XMSS comprise the CNSA Suite 2.0.

The NSA is authoring a set of RFCs, including this one, to provide updated guidance for using the CNSA 2.0 Suite in certain IETF protocols. These RFCs can be used in conjunction with other RFCs and cryptographic guidance (e.g., NIST Special Publications) to properly protect Internet traffic and data-at-rest for U.S. Government National Security Systems.

4. CNSA 2.0 Compliance and Interoperability

As stated in Section 2, all references to "CNSA" in this document refer to CNSA 2.0 [annccnsa], unless stated otherwise.

In some situations, servers or clients normally configured for CNSA compliance may have to interoperate with non-compliant clients and servers. Such servers and clients MUST be configured such that CNSA compliance is preferred even if it is not intended or expected. CNSA-compliant implementations MUST present CNSA compliant algorithms

first in the appropriate extensions, and CNSA-compliant algorithms MUST be used when supported by the peer. If any aspect of CNSA compliance is not achieved, the connection is not CNSA compliant.

If interoperability with non-CNSA-compliant clients or servers is not intended, then the session MUST fail if any aspect of CNSA compliance is not achieved.

If TLS version 1.2 or lower is negotiated, the connection can not be CNSA compliant (Section 4.2).

4.1. CNSA 2.0 Suite

CNSA requires the following algorithms for use in TLS 1.3:

Encryption:

AES [AES] with 256-bit keys, operating in Galois Counter Mode (GCM) [GCM]

Key Establishment:

ML-KEM-1024 [FIPS203] id-alg-ml-kem-1024 OBJECT IDENTIFIER ::= { kems 3 }

Digital Signature

ML-DSA-87 [FIPS204] id-ml-dsa-87 OBJECT IDENTIFIER ::= { sigAlgs 19 }

[ED NOTE: NIST OIDs are written above as placeholders until the TLS registry is updated with identifiers.]

CNSA requires the use of SHA-384 [SHS] for the HMAC-based Key Derivation Function (HKDF) in TLS 1.3.

[ED NOTE: CNSA 2.0 also allows SHA-512 to be used in HKDFs, but at the time of writing, only SHA-384 is supported by TLS 1.3. If SHA-512 becomes available as an option in the TLS registry, CNSA allows such a cipher suite using SHA-512 to be compliant.]

4.2. TLS Version

CNSA TLS servers and clients MUST negotiate TLS version 1.3 [RFC8446] when establishing a CNSA-compliant connection. CNSA TLS clients and CNSA TLS servers MUST NOT negotiate TLS versions prior to TLS 1.3 in a CNSA-compliant mode.

[ED NOTE: No new features will be approved for TLS 1.2 by IETF, including support for quantum-resistant cryptography.]

4.2.1. "supported_versions" Extension

CNSA TLS clients connecting to CNSA TLS servers MUST include the “supported_versions” extension in the ClientHello (Section 4.2.1 of [RFC8446]) and list TLS 1.3 version value (0x0304) first in the extension. CNSA servers establishing a CNSA-compliant connection MUST select TLS 1.3.

5. Key Exchange Messages

This section details requirements for CNSA 2.0-compliant algorithm negotiation in TLS 1.3.

5.1. Cipher Suite Negotiation

The CNSA TLS client MUST offer the following cipher suite in the ClientHello:

TLS_AES_256_GCM_SHA384 {0x13, 0x02} (Appendix B.4 [RFC8847]

This CNSA cipher suite MUST be the first (most preferred) cipher suite in the ClientHello message and in the extensions.

A CNSA TLS server MUST select this cipher suite if offered by the client.

Any cipher suite other than TLS_AES_256_GCM_SHA384 offered by the client MUST NOT be accepted by a CNSA TLS server establishing a CNSA-compliant connection.

[ED NOTE: CNSA 2.0 also allows SHA-512 to be used in HKDFs, but at the time of writing, only SHA-384 is supported by TLS 1.3. If SHA-512 becomes available as an option in the TLS registry, CNSA allows such a cipher suite using SHA-512 to be compliant.]

5.2. KEM Requirements

CNSA 2.0 requires the use of ML-KEM-1024 for key establishment in TLS.

5.2.1. "supported_groups" Extension

The CNSA TLS client MUST offer the following value in named_group_list of the “supported_groups” extension (Section 4.2.7 [RFC8446] of the ClientHello:

ML-KEM-1024

[ED NOTE: -draft8447bis will replace "Elliptic curve groups" in the registry with "unallocated" to allow for KEMs. We await draft-connolly-tls-mlkem-key-agreement to assign an identifier for ML-KEM-1024.]

ML-KEM-1024 MUST be the first (most preferred) item in named_group_list.

The CNSA TLS server MUST select ML-KEM-1024 if it is included in the “supported_groups” extension sent by the CNSA TLS client in the ClientHello.

Any algorithm other than ML-KEM-1024 offered by the client MUST NOT be accepted by a CNSA TLS server establishing a CNSA-compliant connection.

5.2.2. "key_share" Extension

In order to facilitate use of a KEM, the public key and ciphertext are sent via the “key_share” extension in the ClientHello and ServerHello, respectively [I-D.connolly-tls-mlkem-key-agreement]. Specifically, the public key, generated from the ML-KEM-1024 KeyGen algorithm is sent by the client in the KeyShareClientHello value of the extension_data field in the “key_share” extension. The KEM ciphertext generated from ML-KEM-1024 Encaps algorithm is then transmitted from the server back to the client via the KeyShareServerHello value of the extension_data field of the extension.

[ED NOTE: Much of this guidance is written in [I-D.connolly-tls-mlkem-key-agreement] but as it’s in draft form, we rewrite here for clarity.]

The “key_share” extension in the CNSA TLS client ClientHello MUST contain the ML-KEM-1024 public key (Section 4.2, [I-D.connolly-tls-mlkem-key-agreement]).

The ML-KEM-1024 public key must be the first (most preferred) key in the list of key shares.

The CNSA TLS server MUST select the ML-KEM-1024 key share for key establishment, and the “key_share” extension in the CNSA TLS server ServerHello MUST contain the ML-KEM-1024 ciphertext associated to the public key provided in the ClientHello (Section 4.2, [I-D.connolly-tls-mlkem-key-agreement]).

6. Authentication Messages

CNSA TLS clients MUST require the server to authenticate itself. In all cases, the CNSA TLS client MUST authenticate the server using X.509 certificates; PSK-based authentication may be used in addition to certificate-based authentication as detailed in Section 7.

The CNSA TLS server MAY also authenticate the client as needed by the specific application. If mutual authentication is desired, X.509 certificates MUST be used in all cases.

6.1. "signature_algorithms" Extension

A CNSA TLS client MUST offer the following value in the “signature_algorithms” extension of the ClientHello:

ML-DSA-87

The ML-DSA-87 algorithm must be the first (most preferred) value in the extension.

Any signature algorithm values offered other than ML-DSA-87 MUST NOT be accepted by a CNSA TLS server establishing a CNSA-compliant connection.

6.2. "signature_algorithms_cert" Extension

The “signature_algorithms_cert” extension is not required for a CNSA-compliant connection, as ML-DSA-87 is the only allowed signature algorithm that can be used for both signatures in the certificate and in the CertificateVerify message under CNSA compliance.

If the “signature_algorithms_cert” extension is included, the CNSA TLS client MUST offer the following value first in the “supported_signature_algorithms” field of the extension:

ML-DSA-87

Any signature algorithm offered other than ML-DSA-87 MUST NOT be accepted by a CNSA TLS server establishing a CNSA-compliant connection.

6.3. CertificateRequest Message

The CNSA TLS server MAY authenticate the client as needed by the specific application.

If the CNSA TLS server requires mutual authentication, it MUST authenticate the client using X.509 certificates.

The “signature_algorithms” extension sent by the CNSA TLS server in the CertificateRequest message MUST include:

ML-DSA-87

The “signature_algorithms_cert” extension is not required, but if it is included by the CNSA TLS server in the CertificateRequest message, then it MUST include:

ML-DSA-87

ML-DSA-87 MUST be the first (most preferred) algorithm in the “signature_algorithms” and “signature_algorithms_cert” extensions.

6.4. Certificate Selection

Certificates sent by the CNSA TLS server (and CNSA TLS client, when required) MUST be signed by ML-DSA-87.

If the CNSA TLS server sends a CertificateRequest message to the client, the client MUST respond with a valid X.509 certificate suitable for authenticating the client. If the CNSA TLS client sends an empty Certificate message, the CNSA TLS server MUST abort the handshake (Section 4.4.2.4 of [RFC8446]). Also, if some aspect of the certificate chain was unacceptable (e.g., it was not signed by a known, trusted CA), the server MUST abort the handshake.

6.5. CertificateVerify Message

A CNSA TLS client or CNSA TLS server MUST use ML-DSA-87 when sending the CertificateVerify message. CNSA TLS clients or servers MUST only accept ML-DSA-87 in the CertificateVerify message when establishing a CNSA-compliant connection.

7. External Pre-Shared Keys

RFC 8773 [I-D.ietf-tls-8773bis] specifies an extension that allows an external PSK to be used for authentication in addition to (and not in lieu of) certificate-based authentication during the initial handshake. The PSK also contributes to the TLS 1.3 key schedule (Section 7.1, [RFC8446]).

A CNSA TLS client that supports RFC 8773 MUST include the “tls_cert_with_extern_psk” extension (Section 4, [I-D.ietf-tls-8773bis]) in the ClientHello message if it desires an

external PSK to be used for authentication with certificate-based authentication. This extension is accompanied by the “key_share”, “psk_key_exchange_modes”, and “pre_shared_key” extensions defined in (Section 4.2.9 and 4.2.11, respectively, of [RFC8446]).

The “psk_key_exchange_modes” extension MUST NOT include psk_ke mode. The “psk_key_exchange_modes” extension MUST be psk_dhe_key using ML-KEM-1024.

[Ed Note: psk_dhe_key is defined in Section 4.2.9 of RFC 8446 as PSK with (EC)DHE key establishment. We require use of ML-KEM-1024 in the “key_share” extension, as detailed in Section 5.2.2., but must still use psk_dhe_key.]

PSKs shall be at least 256 bits in length, and generated from a NIST approved random bit generator that supports 256-bits of entropy [SP800-90c].

If the CNSA TLS server recognizes the “tls_cert_with _extern_psk” extension and possesses at least one of the external PSKs offered by the client in the “pre_shared_key” extension in the ClientHello, then the CNSA TLS server MUST select a CNSA compliant PSK and respond with the “tls_cert_with_extern_psk”, “key_share”, and “pre_shared_key” extensions in the ServerHello message (Section 4, [I-D.ietf-tls-8773bis]).

8. Resumption

A CNSA TLS server MAY send a CNSA TLS client a NewSessionTicket message to enable resumption. A CNSA TLS client MUST request “psk_dhe_ke” via the “psk_key_exchange_modes” extension in the ClientHello to resume a session. The “supported_groups” and “key_share” extensions MUST comply with those detailed in Section 5.2.1 and Section 5.2.2 of this document, respectively.

9. Certificate Status

A CNSA TLS server (and CNSA TLS client, if client authentication is required) MUST provide certificate status information either via a Certificate Revocation List (CRL) distribution points or by the Online Certificate Status Protocol (OCSP) (with or without stapling) For details on CNSA requirements for these methods, see [I-D.jenkins-cnsa2-pkix-profile].

10. "early_data" Extension

A CNSA TLS client or server MUST NOT include the “early_data” extension.

11. Security Considerations

Most of the security considerations for this document are described in [RFC8446], [I-D.ietf-tls-8773bis]. Additional security considerations for the use of ML-KEM in TLS 1.3 can be found in [I-D.connolly-tls-mlkem-key-agreement].

In order to meet the goal of a consistent security level for the entire cipher suite, CNSA TLS implementations MUST only use the algorithms listed in this document. If this is not the case, security may be weaker than is required.

As noted in TLS version 1.3 [RFC8446], TLS does not provide inherent replay protections for early data. For this reason, this profile forbids the use of early data.

12. IANA Considerations

This document has no IANA considerations.

13. References

[AES] National Institute of Standards and Technology, "Announcing the ADVANCED ENCRYPTION STANDARD (AES)", FIPS 197, DOI 10.6028/NIST.FIPS.197, November 2001, <https://nvlpubs.nist.gov/nistpubs/fips/NIST.FIPS.197.pdf>.

[annccnsa] National Security Agency, "Announcing the Commercial National Security Algorithm Suite 2.0", September 2022, <https://media.defense.gov/2022/Sep/07/2003071834/-1/-1/0/CSA_CNSA_2.0_ALGORITHMS_.PDF>.

[FIPS203] National Institute of Standards and Technology (2024), "Module-Lattice-Based Key-Encapsulation Mechanism Standard", (Department of Commerce, Washington, D.C.), Federal Information Processing Standards Publication (FIPS), NIST FIPS 203, <https://doi.org/10.6028/NIST.FIPS.203>.

[FIPS204] National Institute of Standards and Technology (2024), "Module-Lattice-Based Digital Signature Standard", (Department of Commerce, Washington, D.C.), Federal Information Processing Standards Publication (FIPS), NIST FIPS 204, <https://doi.org/10.6028/NIST.FIPS.204>.

[GCM] Dworkin, M., "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", NIST Special Publication 800-38D, DOI 10.6028/NIST.SP.800-38D, November 2007, <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf>.

[I-D.connolly-tls-mlkem-key-agreement] Connolly, D., "ML-KEM Post-Quantum Key Agreement for TLS 1.3", March 2024, <https://datatracker.ietf.org/doc/draft-connolly-tls-mlkem-key-agreement/01/>.

[I-D.ietf-tls-8773bis] Housley, R., "TLS 1.3 Extension for Using Certificates with an External Pre-Shared Key", July 2024, <https://datatracker.ietf.org/doc/draft-ietf-tls-8773bis/02/>.

[I-D.jenkins-cnsa2-pkix-profile] Jenkins, M. and A. Becker, "Commercial National Security Algorithm Suite Certificate and Certificate Revocation List Profile", January 2025, <https://datatracker.ietf.org/doc/draft-jenkins-cnsa2-pkix-profile/>.

[NSM-10] United States, The White House, "National Security Memorandum on Promoting United States Leadership in Quantum Computing While Mitigating Risks to Vulnerable Cryptographic Systems", NSM 10, May 2022, <https://www.whitehouse.gov/briefing-room/statements-releases/2022/05/04/national-security-memorandum-on-promoting-united-states-leadership-in-quantum-computing-while-mitigating-risks-to-vulnerable-cryptographic-systems/>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8847] Presta, R. and S P. Romano, "Protocol for Controlling Multiple Streams for Telepresence (CLUE)", RFC 8847, DOI 10.17487/RFC8847, January 2021, <https://www.rfc-editor.org/info/rfc8847>.

[SHS] National Institute of Standards and Technology (NIST), "Secure Hash Standard (SHS)", DOI 10.6028/NIST.FIPS.180-4, FIPS PUB 180-4, August 2015, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>.

[SP800-90c] National Institute of Standards and Technology, "Recommendation for Random Bit Generator (RBG) Constructions.", NIST SP 800-90C 4pd, <https://doi.org/10.6028/NIST.SP.800-90C.4pd>.

[SP80059] National Institute of Standards and Technology, "Guideline for Identifying an Information System as a National Security System", Special Publication 59, DOI 10.6028/NIST.SP.800-59, August 2003, <https://csrc.nist.gov/publications/detail/sp/800-59/final>.

Author’s Address

Alison Becker
National Security Agency
Email: aebecke@uwe.nsa.gov