ERICSSON

Date: September 19, 2025

Ericsson AB
Group Function Technology
SE-164 80 Stockholm
SWEDEN

Comments on CNSA 2.0 Draft Profiles

Dear NSA,

Thank you for writing these profiles and making them publicly available. The Suite B and CNSA 1.0 RFCs have been highly valuable for Ericsson, helping us strengthen both our own profiles and the 3GPP profiles. We expect the CNSA 2.0 RFCs to be just as valuable and to further enhance the security of commercial implementations.

IETF protocol specifications and profiles are typically not directly usable for critical infrastructure such as 5G and 6G. For example, adopting BCP 195 as is would have significantly reduced the security of the 3GPP TLS profile in TS 33.210 [9]. Having hardened profiles that explicitly forbid weaker options is therefore highly valuable. We welcome NSA's efforts to strengthen standards and implementations, as well as to ensure compliance with NIST guidelines. We strongly support the publication of CNSA and similar high-security profiles as RFCs.

Below are Ericsson's detailed comments on the CNSA 2.0 draft profiles [1–6]. Overall, the drafts look very good, and with the exception of the IPsec draft, they appear close to being ready for publication.

General comments applicable to multiple drafts

- There are inconsistencies in the title naming conventions. For example, the PKIX draft omits "(CNSA)", the IPsec draft includes "2.0", and the PKIX title ends with "Profile", while the others use "Profile for" or "Profile of".

- *"It is also appropriate for all other US Government systems that process high-value information."*

  The profiles are also important for availability, especially the X.509 profile [1], which underpins firmware and software signing, secure boot, and remote attestation. It would be useful to highlight this aspect. Even when a system does not process classified information, its availability can still be critical. We also consider these profiles valuable for global critical infrastructure in general, including 5G and 6G networks.

- The [cnsafaq] link is not working and should likely be replaced with [7]. Reference [8] appears outdated, as it still mentions CRYSTALS-Kyber and CRYSTALS-Dilithium and does not reference FIPS 203—204. Additionally, the dates in the defense.gov URLs are likely to confuse readers.

- *"With SHA384 (or SHA512), AES-256, and LMS/XMSS, these comprise the CNSA Suite 2.0."*

  According to the CNSA 2.0 FAQ [7], SHA3-384 and SHA3-512 are just as much a part of CNSA 2.0 as LMS and XMSS. Both NIST and CNSA 2.0 [8] use the hyphenated names SHA-384 and SHA-512. It would be useful to inform readers and developers that the newly included SHA-512 is significantly weaker than SHA-384 with respect to length-extension attacks. We recommend SHA-384 over SHA-512. In the future we think both industrial systems and national security systems should use SHAKE256 aligning with ML-KEM and ML-DSA. SHAKE256 offers superior theoretical and practical properties compared to SHA-2 and can often significantly reduce implementation complexity [10], as illustrated, for example, in Section 11 of FIPS 205 [13].

- It would be helpful for all drafts to explicitly state that HashML-DSA is not permitted in CNSA, as specified in [7].

- For ML-DSA, it would be helpful to explicitly clarify whether both hedged and deterministic signing are allowed. In FIPS 204 [12], hedged signing is mandatory to implement and recommended for use, whereas deterministic signing is optional and, in many cases, not recommended.

- *"(CNSA) 2.0 Suite, a cybersecurity advisory"*

  Is CNSA just an advisory? Our understanding is that it constitutes a policy.

- The drafts refer to "systems," "components," "solutions," and "applications." It might be clearer to use fewer terms for consistency.

- We think the CNSA profiles should align with FIPS 204 [12] by using the symbol μ for the message representative. FIPS 204 does not use the term mu, and RFCs can make use of Unicode characters like μ. In Ancient Greek, the letter μ was pronounced /myː/, whereas in Modern Greek it is pronounced /mi/.

- *"The NSA is authoring a set of RFCs"*

  It would be helpful if each draft included references to the other five CNSA 2.0 drafts, ensuring that readers can easily identify and access the complete set. If a draft, such as the PKIX profile, is published before the others, a note could clarify that the remaining drafts are still work in progress.

- The drafts should explicitly require revocation checking and define how frequently it must be performed, as this is critical in high-security systems. We recommend mandating frequent certificate status checks with a clearly specified interval to ensure timely detection of revoked or suspended certificates and preserve system integrity. As an alternative measure, the use of short-lived ephemeral device certificates could also be considered.

- To prevent authentication from being assumed beyond a certificate's expiration, the TLS, IPsec, and SSH drafts should specify that connections SHALL be terminated once the peer's certificate expires. Frequent revocation checking or frequent rekeying with external PSKs could serve as complementary or alternative measures.

- We recommend that the profiles explicitly address the availability properties of each protocol. For instance, a major difference between TLS and DTLS is that in TLS, an attacker can terminate a connection with a single malformed record, whereas DTLS is more vulnerable to handshake amplification attacks. Similar considerations apply to SSH, QUIC, IKEv2, and ESP.

- We recommend that the SSH, TLS, and IPsec profiles explicitly require implementations to support Traffic Flow Security (TFS), i.e., padding and dummy packets. The decision to employ TFS should depend on the specific use case. In certain systems, TFS is required to conceal not only the content of communication but also when and how it occur. TFS can often achieve higher performance and security if implemented at the security protocol level rather than only within the application.

- We recommend that the profiles explicitly mandate that security protocol implementations make certificate chains, CRLs, and OCSP responses available to the application. Implementations sometimes do not even support the application retrieving the peer's certificate chain, which renders both certificate chain validation and identity authentication impossible.

- The X.509DoS study [16] demonstrates several DoS vulnerabilities in mainstream cryptographic libraries via crafted X.509 certificates. Drawing on these findings, and given that CNSA makes extensive use of ASN.1, we recommend that the profiles include a dedicated section on practical security considerations, particularly addressing implementation behavior under malformed or extreme inputs. We suggest that the profiles mandate strict parsing and require implementations to immediately reject any data that is not canonically encoded, as this mitigates a broad class of parsing-based attacks.

- High-security systems require robust auditing. We recommend that the profiles mandate auditable logging of all security-relevant events. For example, a certificate validation failure due to an unavailable CRL should generate a distinct log entry from one caused by an untrusted CA. Such granularity is essential for effective security monitoring and incident response.

- As noted in the comments below, many libraries not only allow the configuration of non-CNSA algorithms but also offer parameter choices that violate NIST requirements. When an implementation permits such configuration, these options MUST be disabled by default and MUST only be changeable by privileged administrators under documented, auditable, and authorized procedures.

Comments on draft-jenkins-cnsa2-pkix-profile-03 [1]

- The heading states "Updates: 8603" but RFC 8603 is not cited in the text. The draft should explicitly explain how it relates to RFC 8603.

- We suggest changing "id-ML-DSA-87" to "id-ml-dsa-87" to align with "id-alg-ml-kem-1024", CSOR [14], [I-D.ietf-lamps-dilithium-certificates], and the CNSA CMC profile [4].

- *"ML-KEM-1024 [FIPS203] for key management"*

  The terms "key establishment keys" or "key encapsulation keys" seem more appropriate in this context. By contrast, the term *"key management"* generally refers to the entire lifecycle of a key, generation, storage, distribution, destruction, etc.

- *"to support interoperability"* and *"when interoperability is desired"*

  Ensuring high security is likely another primary objective.

- *"A ML-DSA-87 signature verification key"*

  Using "key" or "signature key" would be appropriate here, as the key may also be used for signing.

- *"Where an application or implementation makes it more efficient to pre-hash, then the External-mu mechanism allowed by FIPS 204 and described in Section 8 of I-D.ietf-lamps-dilithium-certificates may be used."*

  It is excellent that this is explicitly identified as an approved method. Note that the S/MIME profile [5] uses the term 'pre-hash' in a way that excludes external-μ, which is consistent with FIPS 204. A clearer approach would be to avoid the term pre-hash entirely and instead state explicitly that HashML-DSA is not permitted. For example:

  "Where an application or implementation makes it more efficient to perform hashing externally, the external-μ mechanism described in Step 6 of Algorithm 7 of [FIPS204] and Section 8 of [I-D.ietf-lamps-dilithium-certificates] may be used. HashML-DSA is not permitted."

- *"These OIDs are used to identify both the algorithm associated with the public key (as part of the Subject Public Key Info field) and the signature"*

  The draft should clarify that only id-ml-dsa-87 is used to identify signatures. If ML-KEM-1024 is used for Proof of Possession (PoP) in a manner similar to RFC 6955 [15], a different OID is presumably used. It would be helpful to explicitly explain this distinction.

- It appears that the document also functions as a CNSA 2.0 OCSP profile. This should be stated earlier in the document.

Comments on draft-becker-cnsa2-ssh-profile-02 [2]

- The heading states "Obsoletes: 9212," but RFC 9212 is not cited in the text. The introduction should explicitly state that the document obsoletes RFC 9212.

- *"the server's ephemeral public host key (or certificate)"*

  The host key is (typically) not ephemeral.

- *"that requirement does not in and of itself imply that a given implementation of Secure Shell is suitable for use in NSS."*

  It would be helpful to explain the rationale. We agree that SSH is not well-suited for high-security systems due to its ad-hoc construction, reliance on non-standardized mechanisms, and the frequent use of passwords and raw public keys in many libraries. We consider X.509 to be the only appropriate choice for high-security systems. Similar to the large number of attacks on TLS 1.0–1.2, the Terrapin attack was not surprising.

- *"This document does not preclude implementations that contain algorithms other than those specified in CNSA 2.0."*

  "However, algorithms that are not CNSA compliant must not be negotiated and used in a CNSA-compliant connection."

  We prefer compartmentalization between different security domains and disabling all other algorithms. Allowing additional algorithms introduces potential vulnerabilities, whether through theoretical issues in protocol algorithm negotiation, protocol implementation bugs, or other implementation flaws that could cause a node to treat a CNSA-compliant connection as non-CNSA, and vice versa. Minimizing the attack surface and adopting a "assume breach and minimize impact" approach is highly advisable.

- *"A CNSA compliant connection MUST NOT allow the reuse of ephemeral/exchange values in a key exchange algorithm due to security concerns related to this practice. In accordance with [SP80056A], an ephemeral private key shall be used in exactly one key establishment transaction and shall be destroyed (zeroized) as soon as possible."*

  It is excellent that this issue is explicitly highlighted and strictly forbidden. It would be helpful to elaborate on the underlying security and privacy concerns, as outlined in [17–18, 31–32]. The profile should also refer to NIST SP 800-227 [30], which prohibits the reuse of ephemeral KEM key pairs. The party performing the encapsulation cannot easily detect reuse.

- *"In addition, the AES-GCM invocation counter is incremented mod 2^64"*

  It is good that this omission in RFC 5647 is addressed. Hopefully, all SSH implementations will adhere to this.

- We think the profile should explicitly address rekeying. By default, implementations like OpenSSH, do not comply with RFC 4253, which normatively states that "It is RECOMMENDED that the keys be changed after each gigabyte of transmitted data or after each hour of connection time, whichever comes sooner." We recommend mandating this behavior to limit the impact of key compromise [19]. This also addresses the fact that SSH specifications and implementations theoretically allow users to encrypt more than 2^{64} blocks. Best practice is to never encrypt more than 2^{59} AES blocks [21–23].

- *"as described in SECTION"*

- *"Where possible, public keys SHOULD be validated with certificates"*

  It would be good to just state "Public keys SHOULD be validated with X.509 certificates". This would put pressure on SSH implementations to support X.509, which is the only suitable option for high-security use cases.

- *"Users MAY be authenticated using passwords"*

  "If authenticating by passwords, it is essential that passwords have sufficient entropy to protect against dictionary attacks."

  This option offers a security level that appears misaligned with the rest of CNSA, as human-created passwords typically provide only about 20–40 bits of entropy. We therefore recommend forbidding password-based authentication.

Comments on draft-becker-cnsa2-tls-profile-02 [3]

- The document should explicitly explain how it relates to RFC 9151.

- *"CNSA-compliant implementations MUST present CNSA compliant algorithms"*

  "If TLS version 1.2 or lower is negotiated"

  As stated in the comment on the SSH profile, we think it is preferable to disable all non-CNSA algorithms as well as TLS 1.2.

- Including an ML-KEM-1024 key share in the TLS handshake may lead to compatibility issues with legacy servers that cannot correctly handle large key shares, large ClientHello messages, or the fragmentation of ClientHello messages across multiple TCP segments. Because the profile explicitly discusses interoperability with both CNSA 1.0 and non-CNSA servers, it is important to highlight the potential impact on deployments that rely on older TLS implementations.

- Section 4.1 ("CNSA 2.0 Suite") and Section 5 appear to contain largely overlapping information. Moreover, {0x13, 0x02} is not limited to AES-GCM as described in Section 4.1. We suggest removing Section 4.1.

- Section 5.2.2 appears to include substantial internal duplication, which could be streamlined for clarity.

- *"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."*

  The same requirement should apply to the client.

- *"A CNSA TLS client or server MUST NOT include the "early_data extension"*

  "The "psk_key_exchange_modes" extension MUST NOT include psk_ke mode"

We welcome that the profile disallows early data and requires psk_dhe_ke. Having a profile to cite for these requirements is valuable.

- The DTLS 1.3 section appears to duplicate much but not all of the earlier content. It would be preferable to list only the changes relative to Sections 4–9.

- The profile should clarify whether QUIC is permitted. From a security and profiling perspective, TLS 1.3, DTLS 1.3, and QUIC are mostly equivalent, as they all employ the TLS 1.3 handshake. The main differences lie in availability. Since QUIC is required for HTTP/3, we recommend that the TLS profile explicitly state that it also applies to QUIC, highlighting any relevant differences.

- *"for the entire cipher suite"*

  The term "cipher suite" here appears to be used in a broader sense than TLS 1.3 cipher suites, which makes it confusing.

- *"TLS does not provide inherent replay protections for early data. For this reason, this profile forbids the use of early data."*

  The profile should also state that DTLS replay protection MUST be used. Unfortunately, the DTLS 1.3 standard and some implementations allows replay protection to be turned off. We think replay protection should be a strong requirement. Users and developers expect replay protection and higher layer protocols are often designed with the expectation that the security protocol provides replay protection. Systems without replay protection often leads to surprising attacks and are hard to analyze. If an upper layer was designed with the expectation of replay protection in a lower layer, using a security protocol without replay protection in the lower layers can compromise confidentiality, integrity, and availability in the higher layer, i.e., the whole infosec CIA triad. Practical and serious vulnerability due to the lack of replay protection has been common in both standardized and proprietary systems [20].

- As correctly identified by the SSH and IPsec profiles: *"A CNSA compliant connection MUST NOT allow the reuse of ephemeral/exchange values in a key exchange algorithm due to security concerns related to this practice. In accordance with [SP80056A], an ephemeral private key shall be used in exactly one key establishment transaction and shall be destroyed (zeroized) as soon as possible."*

  These security concerns apply equally to TLS 1.3. We strongly recommend adding the same text to the TLS profile. In fact, since TLS is much more commonly used to connect to a wide range of peers, this requirement is even more important for TLS than for IPsec. Some of the security and privacy risks with reuse are explained in [17–18, 31–32]. NIST SP 800-227 [30] forbids reuse of ephemeral KEM key pairs. We strongly recommend using implementations that do not violate NIST requirements. Adopting an "assume breach and minimize impact" approach is highly advisable. NIST SP 1800-37 [31] discourages the reuse of key shares, i.e., the use of static keys, as a visibility/intercept mechanism in TLS 1.3.

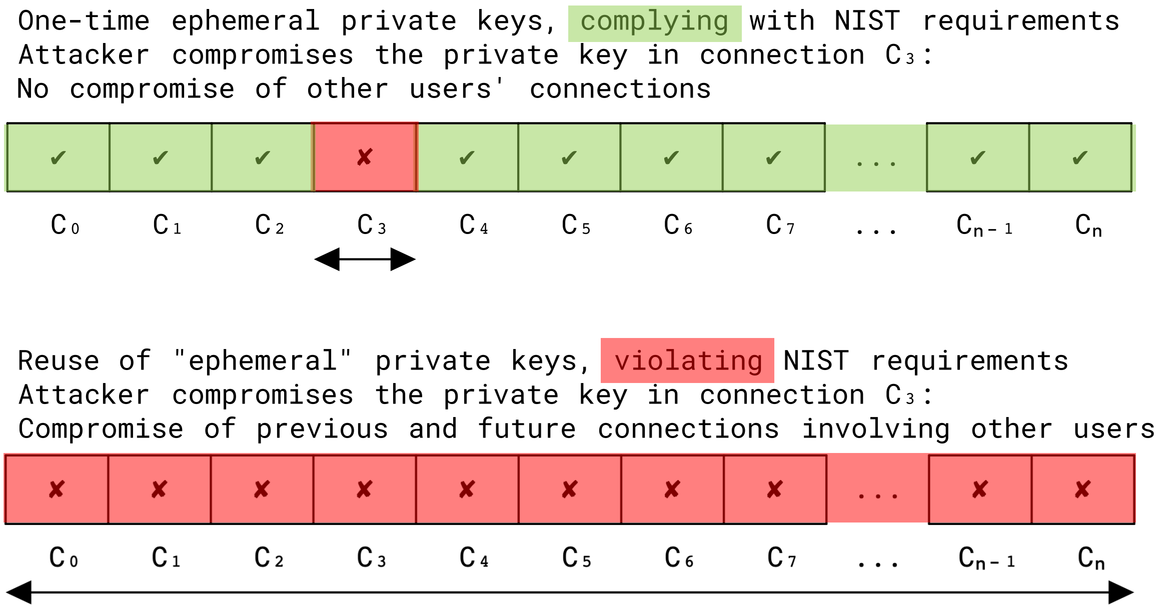
One-time ephemeral private keys, complying with NIST requirements
Attacker compromises the private key in connection C₃:
No compromise of other users' connections
✓ ✓ ✓ ✘ ✓ ✓ ✓ ✓ ... ✓ ✓
C₀ C₁ C₂ C₃ C₄ C₅ C₆ C₇ ... Cₙ₋₁ Cₙ
Reuse of "ephemeral" private keys, violating NIST requirements
Attacker compromises the private key in connection C₃:
Compromise of previous and future connections involving other users
✘ ✘ ✘ ✘ ✘ ✘ ✘ ✘ ... ✘ ✘
C₀ C₁ C₂ C₃ C₄ C₅ C₆ C₇ ... Cₙ₋₁ Cₙ

Figure 1. Impact of session key compromise on two servers connected to multiple users: one server complies with NIST requirements by not reusing ephemeral keys, while the other violates this requirement.

- *"In some situations, clients or servers configured for CNSA compliance may have to interoperate with non-compliant clients and servers."*

  Reusing "ephemeral" keys between CNSA and non-CNSA connections seems highly insecure, as it breaks both session key independence and compartmentalization. Past incidents have shown that reuse of "ephemeral" keys, combined with implementation flaws, can lead to severe practical vulnerabilities in which attackers recover the so-called "ephemeral" private key, enabling full compromise of sessions between legitimate parties. In such a scenario, a non-CNSA website, for example egotisticalpanda.net or egotisticalbear.net, controlled by a hostile nation could passively eavesdrop on CNSA connections or inject malicious traffic into them, see Figure 1. We strongly recommend compartmentalization between red and black networks and that all reuse of ephemeral keys be explicitly forbidden.

- We strongly think the profile should explicitly address rekeying. While RFC 8446 specifies that implementations SHOULD perform a key update before reaching $2^{24.5}$ full-size records equaling $2^{34.5}$ blocks, the key update mechanism is much weaker than the mechanism required in the SSH profile, and does not protect against key leakage. Furthermore RFC 8446, and to our understanding OpenSSL, does not mandate rekeying before reaching 2^{64} records equaling up to 2^{74} blocks, or theoretically even more in the case of very large TLS records [24]. In OpenSSL, compliance to any rekeying limit is left entirely to the application, which is poor security design. We recommend that the CNSA profile mandate that TLS 1.3 implementations enforce a 2^{59} block limit [21–23]. A compliant implementation should be required to automatically trigger a key update or terminate the connection before the limit is reached, placing the enforcement burden on the protocol implementation, not the application developer. Frequent rekeying with ephemeral asymmetric key exchange, similar to the requirements for SSH and IPsec, and which can be achieved with resumption, should be recommended.

Comments on draft-jenkins-cnsa2-cmc-profile-01 [4]

- *"This profile may be further tailored by specific communities to meet their needs."*

  Should be clarified that the signature and hash algorithms cannot be changed.

- See the PKIX comments for discussion of the term *"key management"*

- *"the RA and/or CA whose certificate is being managed is considered to be the end-entity"*

  It would be helpful to clarify to the reader that RFC 5272 appears to use a different definition of "end entity" than RFC 5280, which states that "End entity certificates are issued to subjects that are not authorized to issue certificates."

- *"unless an appropriate signature certificate does not yet exist, such as during initial enrollment."*

  It would be helpful to note that in such cases, another suitable security measure must be used.

- *"CAs conforming to this document MUST ensure that only the permitted signature, hash, and MAC algorithms described throughout this profile"*

  We cannot find any MAC algorithms described in the profile.

- We recommend that the profile explicitly address the generation and storage of private keys within HSMs. Using HSMs for key generation, storage, and usage helps mitigate the risk of key compromise and aligns with best practices for high-assurance cryptographic implementations.

- *"FIPS 186-3 [FIPS186]"*

  [FIPS186] is FIPS 186-4, which has been superseded by FIPS 186-5 [25].

Comments on draft-becker-cnsa2-smime-profile-01 [5]

- The heading states "Obsoletes: 9212," but it should reference RFC 8755. The introduction should also explicitly mention that the document obsoletes RFC 8755.

- *"CMS values are generated using ASN.1 [X208], the Basic Encoding Rules (BER) [X209], and the Distinguished Encoding Rules (DER) [X509]."*

  X.680 [26] and X.690 [27] appear to be the appropriate references for ASN.1, BER, and DER. If they are not suitable, this should be stated explicitly, as anyone searching for X.208 and X.209 will see that these have been withdrawn and superseded. CNSA should consider restricting encodings to DER or CER, as full BER support does not appear necessary for CMS and could introduce security vulnerabilities.

- *"All implementations MUST use SHA-384 or SHA-512 for hashing and AES-GCM for encryption"*

  AES-GCM is only used for encrypting content, while AES-KWP is used for encrypting keys.

- *"Note that [I-D.ietf-lamps-cms-ml-dsa] only specifies use of the pure mode (not pre-hash) of ML-DSA in CMS"*

  Note that the CNSA PKIX profile [1] uses the term 'pre-hash' in a way that includes external-µ. However, FIPS 204 describes external-µ as being pure and not pre-hash, which also creates confusion. The best approach is likely to avoid using the term 'pre-hash' altogether and instead state explicitly that HashML-DSA is not permitted.

- *"The "message digest" supplied to the signature algorithm is the entire message."*

  This accurately describes pure ML-DSA, but it can be confusing in the context of S/MIME. When signedAttrs are present in S/MIME, which is typically the case, the signature is calculated over the signed attributes, including the messageDigest attribute, rather than over the entire S/MIME message.

- *"CNSA-compliant implementations MUST NOT support the user keying material (ukm) field"*

  It would be helpful to clarify whether this is required for interoperability or security purposes. The CNSA TLS [3] and IPsec [6] profiles permits combining ML-KEM-1024 with user keying material in the form of external PSKs.

- *"A CNSA compliant implementation MUST use SHA-384 or SHA-512 for KDF computation"*

  It would be helpful to state explicitly whether all of HKDF, KDF2, and KDF3 are allowed. Since [I-D.ietf-lamps-cms-kyber] mandates support for HKDF, we recommend CNSA to mandate the use of HKDF and prohibit KDF2 and KDF3. KDF2 and KDF3 have weaker properties, especially when used with SHA-512, and are harder to analyze.

  Allowing all six combinations of SHA-384, SHA-512, HKDF, KDF2, and KDF3 is likely to cause interoperability issues. Recipients should be required to support all algorithms that the sender is permitted to use.

- *"The initialization vector (aes-nonce) MUST be generated in accordance with Section 8.2 of [SP80038D]. AES-GCM loses security catastrophically if a nonce is reused with a given key on more than one distinct set of input data. Therefore, a fresh content-authenticated encryption key MUST be generated for each message."*

  It is excellent that this issue is explicitly highlighted and addressed. However if a fresh key is generated for each message, all initialization vectors comply with SP 800-38D. Are all nonce lengths allowed in CNSA? SP 800-38D recommends that implementations only support 12-byte nonces. Since the CNSA S/MIME profile requires that a fresh encryption key MUST be generated for each message, we recommend mandating the use of 12-byte nonces. "content-authenticated encryption key" should likely be "content-encryption key"

- For messages at high classification levels, certificate status checking via CRL or OCSP should be mandatory prior to plaintext release.

- [GCM] and [SP80038D] refer to the same document. The link in [SP80038D] is broken because it includes an extraneous '>' at the end.

Comments on draft-guthrie-cnsa2-ipsec-profile-01 [6]

- The heading states "Obsoletes: 9212," but RFC 9212 is not cited in the text. The introduction should explicitly state that the document obsoletes RFC 9212.

- *"Post-Quantum Pre-Shared Key"*

  We recommend that neither CNSA nor the IETF use this term. All pre-shared keys are inherently resistant to quantum attacks, and the term post-quantum is typically applied to algorithms rather than to keys themselves.

- It would be helpful if the profile specified early on, at a high level, exactly how many key exchanges are mandated or allowed and the general contents of each.

- *"The key exchange performed in IKE_SA_INIT can use any of the following algorithms: 384-bit random ECP group, 3072-bit MODP group, 4096-bit MODP group."*

  "The CNSA 1.0 key establishment algorithms are permitted instead of, e.g., a smaller quantum resistant algorithm because they enable backwards compatibility with CNSA 1.0-compliant implementations of IPsec during the transition"

  According to the anticipated CNSA 2.0 timeline [7], network equipment is required to support and prefer ML-KEM-1024 by 2026. In the future, when backward compatibility is no longer required, it would be good to allow ML-KEM-512, ML-KEM-768, or NONE. Future CNSA implementations should be able to disable ECP and MODP groups. ECP and MODP groups are much slower than ML-KEM-1024 with 4096-bit MODP being around 500 times slower than ML-KEM-1024. Curve25519 and Curve448 are the only appropriate options for hybridizing ML-KEM [10].

- *"If ML-KEM-1024 were used in the IKE_SA_INIT exchange, the sizes of its public key and ciphertext would cause the initiator and responder messages to exceed the typical path MTU and necessitate in IP-level fragmentation"*

  While this may hold true for many networks, jumbo frames are widely supported in internal datacenter networks. We do not believe that CNSA should prohibit the use of ML-KEM-1024 in IKE_SA_INIT for networks of this type or when IKE is transported over TCP [33]. If this prohibition is intended, effectively mandating hybridization, it should be stated clearly and explicitly. We propose the following text that allows for optimal performance where conditions permit:

"Implementations transporting IKE over UDP and not performing Path MTU (PMTU) discovery SHOULD avoid using ML-KEM-1024 in the IKE_SA_INIT exchange on networks where the PMTU is unknown or restricted. However, when IKE is transported over TCP and on networks where a sufficient PMTU is guaranteed, implementations MAY use ML-KEM-1024 in the IKE_SA_INIT."

- *"use of ML-DSA vs. ExternalMu-ML-DSA"*

  We do not believe this terminology is appropriate and it is likely to confuse readers. "External-µ" is simply a variant of pure ML-DSA. We suggest phrasing it as: "internal-µ and external-µ variants of ML-DSA; see Step 6 of Algorithm 7 in FIPS 204."

- We agree that it is beneficial to require CERTREQ and most modern IKEv2 implementations support it.

- *"While IKEv2 allows for the reuse of ephemeral Diffie-Hellman private keys [RFC7296], Section 2.12, there are security concerns related to this practice. In order to address such concerns, [I-D.ietf-ipsecme-ikev2-mlkem] requires ephemeral keys to be generated per connection. Moreover, this profile REQUIRES CNSA 2.0-compliant IPsec implementations to align with [SP80056A]. In particular, an ephemeral private key MUST be used in exactly one key establishment transaction and MUST be destroyed (zeroized) as soon as possible. Any shared secret derived from key establishment MUST also be destroyed (zeroized) immediately after its use."*

  It is excellent that this issue is explicitly highlighted and strictly forbidden. It would be helpful to elaborate on the underlying security concerns, for example as discussed in [17–18, 31–32]. The profile should also refer to NIST SP 800-227 [30], which prohibits the reuse of ephemeral KEM key pairs.

- We welcome the discussion of AES-GCM-SIV; however, we believe it is primarily useful in non-IKEv2 deployments. Nonce reuse is particularly dangerous in non-IKEv2 ESP deployments that are transitioning from 3DES-CBC or AES-CBC to AES-GCM. While key reuse between SAs might be acceptable with CBC, it is not with GCM, and this has led to serious practical vulnerabilities in for example 3GPP IMS implementations for VoLTE and VoNR when using the algorithms "aes-gcm" and "aes-gmac." Ericsson proposed that support for "aes-gcm" and "aes-gmac" should be prohibited, as it is in our products, but this proposal was blocked by other vendors. It remains unclear whether this was due to a lack of understanding that AES-GCM with guaranteed key-nonce collisions offers virtually no security, or whether vendors had disabled the mandatory to support backup algorithms AES-CBC and HMAC-SHA1 in violation of the standard.

- *"ML-KEM-1024 has Key Exchage Method value "TBD37" [EDNOTE: Will be assigned by IANA when [I-D.ietf-ipsecme-ikev2-mlkem] is published]."*

  The value 37 has already been assigned by IANA [28] and is already supported by many IPsec implementations.

- *"This profile strengthens normative key generation, encapsulation, and decapsulation guidance from [I-D.ietf-ipsecme-ikev2-mlkem]"*

It is excellent that this is highlighted. We are strongly opposed to any implementations or IETF specifications that violate NIST requirements. We are likely to object to the publication of [I-D.ietf-ipsecme-ikev2-mlkem] in its current form.

- ENCR_AES_GCM_16, as specified in RFC 4106 [29], relies on non-standardized cryptography. The CNSA profile should clearly state that implementations MUST comply with NIST SP 800-38D.

- The profile should state that replay protection MUST be used. Unfortunately, IPSec standards and many implementations allow replay protection to be turned off. As stated in the comment on the TLS profile, we think replay protection should be a strong requirement.

- We recommend that the profile explicitly address the rekeying of both IKE and Child SAs. Most IPsec implementations allow users to configure settings that theoretically allow users to encrypt more than 2^{64} blocks. Best practice is to never encrypt more than 2^{59} AES blocks [21–23]. The profile should require the use of Perfect Forward Secrecy (PFS) for all Child SAs, define explicit rekeying intervals based on both time and data volume, and clearly state that these limits MUST be enforced by the IPsec implementation. For example, CNSA could align with the ANSSI guidance of rekeying "every hour and every 100 GB of data, in order to limit the impact of a key compromise" [19].

John Preuß Mattsson,
Expert Cryptographic Algorithms and Security Protocols, Ericsson
On behalf of the Ericsson Cryptography Team

References

[1] Commercial National Security Algorithm Suite Certificate and Certificate Revocation List Profile
https://datatracker.ietf.org/doc/html/draft-jenkins-cnsa2-pkix-profile-03

[2] Commercial National Security Algorithm (CNSA) Suite Profile for SSH
https://datatracker.ietf.org/doc/html/draft-becker-cnsa2-ssh-profile-02

[3] Commercial National Security Algorithm (CNSA) Suite Profile for TLS 1.3
https://datatracker.ietf.org/doc/html/draft-becker-cnsa2-tls-profile-02

[4] Commercial National Security Algorithm (CNSA) Suite Profile of Certificate Management over CMS
https://datatracker.ietf.org/doc/html/draft-jenkins-cnsa2-cmc-profile-01

[5] Commercial National Security Algorithm (CNSA) Suite Profile for Secure/Multipurpose Internet Mail Extensions (S/MIME)
https://datatracker.ietf.org/doc/html/draft-becker-cnsa2-smime-profile-01

[6] Commercial National Security Algorithm (CNSA) Suite 2.0 Profile for IPsec
https://datatracker.ietf.org/doc/html/draft-guthrie-cnsa2-ipsec-profile-01

[7] The Commercial National Security Algorithm Suite 2.0 and Quantum Computing FAQ
https://media.defense.gov/2022/Sep/07/2003071836/-1/-1/0/CSI_CNSA_2.0_FAQ_.PDF

[8] Announcing the Commercial National Security Algorithm Suite 2.0
https://media.defense.gov/2025/May/30/2003728741/-1/-1/0/CSA_CNSA_2.0_ALGORITHMS.PDF

[9] Network Domain Security (NDS); IP network layer security
https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2279

[10] Ericsson comments on NIST SP 800-227 (Key-Encapsulation Mechanisms)
https://csrc.nist.gov/files/pubs/sp/800/227/ipd/docs/sp800-227-ipd-public-comments-received.pdf

[11] Module-Lattice-Based Key-Encapsulation Mechanism Standard
https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.pdf

[12] Module-Lattice-Based Digital Signature Standard
https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.pdf

[13] Stateless Hash-Based Digital Signature Standard
https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.205.pdf

[14] Computer Security Objects Register
https://csrc.nist.gov/projects/computer-security-objects-register/algorithm-registration

[15] Diffie-Hellman Proof-of-Possession Algorithms
https://datatracker.ietf.org/doc/html/rfc6955

[16] X.509DoS: Exploiting and Detecting Denial-of-Service Vulnerabilities in Cryptographic Libraries using Crafted X.509 Certificates
https://www.usenix.org/system/files/usenixsecurity25-shi-bing.pdf

[17] ML-KEM is Great! What's Missing? (Paper)
https://csrc.nist.gov/csrc/media/Events/2025/workshop-on-guidance-for-kems/documents/papers/ml-kem-is-great-paper.pdf

[18] ML-KEM is Great! What's Missing? (Slides)
https://csrc.nist.gov/csrc/media/Presentations/2025/ml-kem-is-great/images-media/ml-kem-is-great.pdf

[19] Recommendations for securing networks with IPsec
https://cyber.gouv.fr/sites/default/files/2015/09/NT_IPsec_EN.pdf

[20] Ericsson comments on SP 800-38B (CMAC) and 800-38C (CCM)
https://csrc.nist.gov/csrc/media/Projects/crypto-publication-review-project/documents/initial-comments/sp800-38c-initial-public-comments-2024.pdf

[21] Guide des Mécanismes cryptoraphiques
https://cyber.gouv.fr/sites/default/files/2021/03/anssi-guide-mecanismes_crypto-2.04.pdf

[22] Agreed Cryptographic Mechanisms
https://certification.enisa.europa.eu/document/download/a845662b-aee0-484e-9191-890c4cfa7aaa_en

[23] Collision-Based Attacks on Block Cipher Modes - Exploiting Collisions and Their Absence
https://eprint.iacr.org/2024/1111.pdf

[24] Large Record Sizes for TLS and DTLS with Reduced Overhead
https://datatracker.ietf.org/doc/html/draft-ietf-tls-super-jumbo-record-limit

[25] Digital Signature Standard (DSS)
https://doi.org/10.6028/NIST.FIPS.186-5

[26] Abstract Syntax Notation One (ASN.1): Specification of basic notation
https://www.itu.int/rec/T-REC-X.680-202102-I/en

[27] Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)
https://www.itu.int/rec/T-REC-X.690-202102-I/en

[28] Internet Key Exchange Version 2 (IKEv2) Parameters
https://www.iana.org/assignments/ikev2-parameters/ikev2-parameters.xhtml

[29] The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)
https://www.rfc-editor.org/rfc/rfc4106.html

[30] Recommendations for Key-Encapsulation Mechanisms
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-227.pdf

[31] Addressing Visibility Challenges with TLS 1.3 within the Enterprise
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.1800-37.pdf

[32] Ericsson comments 1800-37A
https://emanjon.github.io/NIST-comments/2023%20-%20SP%201800-37A.pdf

[33] Separate Transports for IKE and ESP
https://datatracker.ietf.org/doc/html/draft-smyslov-ipsecme-ikev2-reliable-transport