ERICSSON

Date: March 7, 2025

Ericsson AB
Group Function Technology
SE-164 80 Stockholm
SWEDEN

# Comments on NIST SP 800-227 Initial Public Draft Recommendations for Key-Encapsulation Mechanisms

Dear NIST,

Thanks for your continuous efforts to produce well-written, user-friendly, and open-access security documents. NIST has done an outstanding job with the standardization of ML-KEM and Post-Quantum Cryptography (PQC) in general. Cryptographers from around the globe have participated and contributed to the PQC project, with discussions being open and public, and all specifications freely accessible. ML-KEM is an excellent general-purpose, single-recipient Key Encapsulation Mechanism (KEM) with exceptional performance in both hardware and software.

We greatly appreciate that the ML-KEM family consists of only three algorithms, offers IND-CCA security, ensures the shared secret is derived from randomness contributed by both parties, and produces a uniformly random shared secret that is immediately usable without requiring additional key derivation. We applaud NIST for mandating the use of SHA-3 in ML-KEM.

Please find below our comments on NIST SP 800-227:

- We think the document should mention that the transition to quantum-resistant cryptography presents an excellent opportunity to reassess outdated algorithms and practices that no longer meet acceptable security standards.

- We strongly support the excellent NIST SP 800-56A requirement that "*An ephemeral private key shall be used in exactly one key-establishment transaction.*" The practice in some security protocols of reusing private keys while still labeling them as ephemeral is deeply problematic and misleading. Users and developers expect that ephemeral keys are used only once and that their security is independent of sessions involving hostile adversaries. The reuse of ephemeral keys, combined with implementation bugs such as the lack of public key validation, has resulted in serious exploitable vulnerabilities. These flaws have allowed attackers to recover the so-called "ephemeral" private key, enabling them to completely compromise sessions between legitimate

parties. Implementation bugs that allow attackers to recover private keys have been well-documented for both ECC and RSA, and similar vulnerabilities are likely for quantum-resistant KEMs. Always assuming breach and minimizing the impact of breach are essential zero-trust principles. Any protocol that reuses private keys should explicitly acknowledge this practice and state that the keys are (semi-)static.

We strongly recommend that the same requirement "*An ephemeral private key shall be used in exactly one key-establishment transaction*" be included in all NIST specifications on KEMs including SP 800-227. ML-KEM is so fast that reusing private keys to save a few CPU cycles is not justifiable. At a minimum, reusing the same ephemeral private key for different users must be strictly forbidden. It is essential that the keying material used for different users is independent.

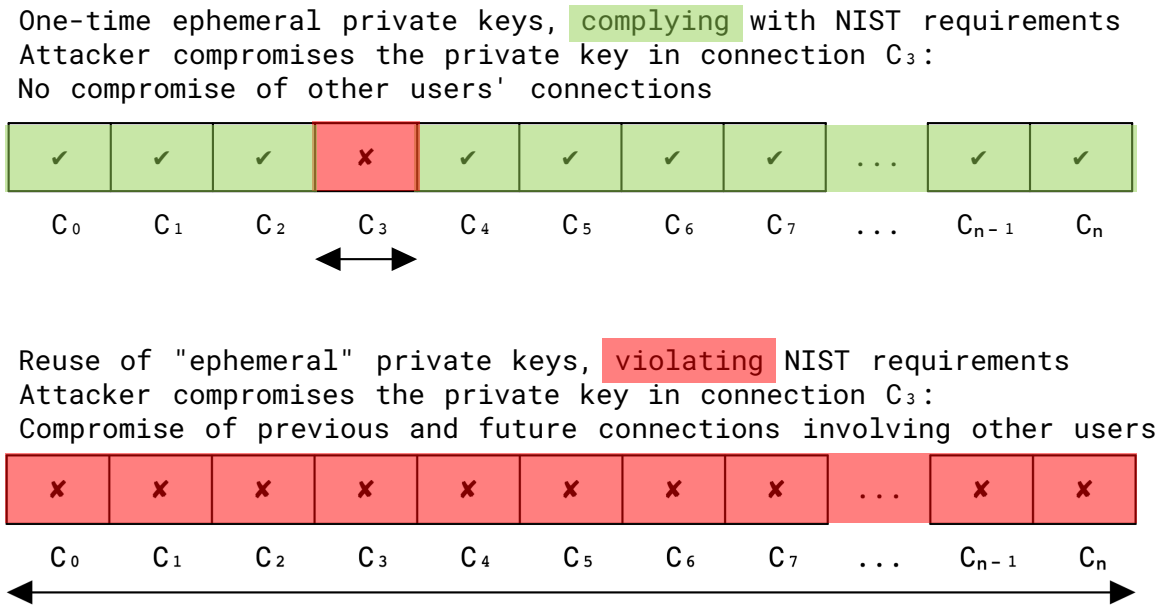


*Figure 1. Impact of session key compromise on two servers connected to multiple users: one server complies with NIST requirements by not reusing ephemeral keys, while the other violates this requirement.*

- We think SP 800-227 should discuss binding properties [1–2], private key storage formats [3], implicit and explicit rejection [4], and anonymity and robustness of KEM-DEM constructions [5]. We think future KEMs should exhibit strong binding properties (e.g., MAL-BIND-K-CT, MAL-BIND-K-PK, etc.), ensure anonymity and robustness in KEM-DEM constructions, use a 32-byte seed as the private key, and be explicitly rejecting. While committing AEADs imposes significant performance costs, KEM binding properties can be achieved with a small performance impact. We do not believe implicit rejection offers practical security benefits in systems—on the contrary, it merely defers the problem, assuming that later steps will properly handle the rejection. This places an additional burden on application correctness, which is disadvantageous. As shown in [4], explicit rejection is as secure as implicit rejection. Moreover, implicit rejection excludes certain binding properties and introduces additional performance overhead.

- We think SP 800-227 should define a (strong) KEM as providing IND-CCA security, deriving the shared secret from randomness contributed by both parties, and outputting a uniformly random shared secret that is immediately usable without additional key derivation. These properties should be required for all future KEMs. Ensuring that all previously described KEMs in literature or standards fit the definition should not be a goal. As the performance of the IND-CCA ML-KEM is truly excellent, we do not think NIST should standardize any IND-CPA KEMs.

- We note that NIST uses the term "cryptographic algorithm" in varying ways. For example, FIPS 197 refers to AES as a single algorithm, whereas SP 800-227 describes ML-KEM as three algorithms, and MACs as a family of algorithms, parameterized by a key. It would be good with more consistent use of the term.

- We think the definition of "negligible" as application-specific in Appendix A.3 is preferred over the strict definition of it as smaller than $2^{-\lambda}$ in Section 2.

- *"Key agreement, KEM, and key transport are all types of key establishment"*

  We think a KEM is quite different from the two other concepts. We suggest rewriting to explain that a KEM can be used for both key agreement and key transport.

- We think SP 800-227 should explain how single-recipient KEMs can be used in multi-recipient settings and which security properties (key transport vs. key agreement) different approaches gives. The use of PKE/KEMs in multi-recipient scenarios is very common. Some examples are S/MIME, Age file encryption, and Signal messaging.

- *"as long as the internal action of the process is hidden from observation"*

  This might make the reader think about security by obscurity. The input, output, and intermediate values need to be hidden from observation, not the process. We suggest reformulating.

- We welcome NIST's plan to allow hybrid shared secrets of the form $S_1 \| S_2 \| \ldots \| S_t$, where at least one of the shared secrets $S_i$ is NIST approved. We anticipate that hybrid shared secrets consisting of more than two components will be relatively common. For instance, [6], authored by the chief cryptographer of the Swedish NCSA, recommends hybrid keying that integrates symmetric keying, post-quantum secure asymmetric keying, and classically secure asymmetric keying. Both TLS 1.3 and IKEv2 support this type of hybrid keying. One concrete example is TLS 1.3 resumption with X25519MLKEM768. Additionally, hybrid keying with X25519/X448, ML-KEM, and BIKE/HQC has been proposed as a conservative construction [7].

- *"A key-encapsulation mechanism (KEM) is a set of algorithms that can be used by two parties under certain conditions to securely establish a shared secret key over a public channel."*

  To securely establish a shared secret it is essential that at least one party on the channel is authenticated; otherwise, you have no idea who you are communicating with. We think it is important that NIST clarifies this. There are a lot of snake-oil companies marketing QKD as impossible to eavesdrop on due to the laws of physics, without acknowledging that its eavesdropping protection ultimately relies on authentication with classical cryptography.

- *"identifier A bit string that is associated with a person, device, or organization"*

  Identifiers are also used for animals, e.g., The National Animal Identification System (NAIS).

- *"key pair A public key and its corresponding private key."*

  Would be good to give encapsulation key and decapsulation key as examples here.

- We think the specification talks too much about old KEMs (RSASVE-KEM and ECDH-KEM) that NIST plans to deprecate 2030. While X25519 and X448 are critical for hybridizing ML-KEM, we see no future need for RSA, FFDH, and Weierstraß curves. Montgomery curves offer superior security and performance and should be the sole choice for ML-KEM hybridization. Hybrids with Montgomery curves have already become the de facto standard for TLS 1.3, DTLS 1.3, QUIC, SSH, Signal, and Rosenpass with widespread deployments. We think NIST should encourage the use of X25519 and X448, as specified in SP 800-186, in hybrid schemes. This aligns with de facto Internet standards. Weierstraß curves has led to a large number of vulnerabilities leading to complete compromise of both confidentiality and integrity in security protocol sessions. The number of implementations that ignores public key validation does not seem to decrease with new examples popping up all the time. We think Weierstraß curves should be phased out in the migration to quantum-resistant cryptography.

  Another factor that has not been discussed nearly enough is the performance of hybridized KEMs. ML-KEM offers exceptional performance with X25519 being the only classical KEM that comes close [8]. In contrast, using any of the standardized Weierstraß curves (NIST P-curves, Brainpool, etc.) significantly reduces performance without offering meaningful benefits. Experience shows that performance is critical for practical deployments—there are many examples of systems disabling cryptography deemed too slow or performing asymmetric key exchanges less frequently than recommended. Best practice for messaging systems is to perform an asymmetric key exchange for each message [9]. Hybridization with Weierstraß curves decreases the performance by several hundred percent compared to standalone ML-KEM and cannot be recommended. We think SP 800-227 should only mention ML-KEM and ECDH-KEMs based on Curve25519 and Curve448 as example KEMs. It is very good that NIST gives X-Wing as the only concrete example of a hybrid PQ/T KEM.

- *"Key confirmation should be used during KEM usage, as it may enhance the security properties of the overall key-establishment process."*

  The document should explain to the reader which security properties can be enhanced. Our experience is that key confirmation is essential for availability. Without key confirmation, a party might send a lot of messages that the recipient cannot process, or a newly issued key card might not work when needed to open a door.

- NIST should take inspiration from [6] and recommend always hybridizing symmetric keying with post-quantum secure asymmetric keying wherever possible. Asymmetrically distributed keys can be refreshed at very frequent intervals to enhance security. 6G, the sixth generation of cellular networks, is expected to incorporate Authentication and Key Agreement (AKA) augmented with quantum-resistant KEMs [10–11], effectively preventing passive eavesdropping on U.S. mobile communications by foreign nation-states that have compromised the symmetric keys.



- *"NIST encourages the use of key combiners that generically preserve IND-CCA security."*

  We think this is a very good recommendation. NIST should state in some specification that *"NIST encourages the use of composite signatures that preserve SUF-CMA security."*

- We applaud NIST for mandating the use of SHA-3 in ML-KEM. SHA-3 is significantly more modern and versatile than SHA-2. Since ML-KEM and ML-DSA are based on SHA-3, it is natural to switch to SHA-3 when migrating to quantum-resistant algorithms instead of sticking to SHA-2. SHA-3 was designed with side-channel security in mind, whereas SHA-2 is significantly harder to protect against side-channels. HMAC, HDKF, MGF, etc. are constructions only needed when the hash function has significant vulnerabilities/issues such as length-extension, failure to behave like a random function, lack of variable-length output, etc. SHA-3 is practically much more secure than SHA-2. Most users do not understand when and how they need to use HMAC/HKDF/MGF. SHA-3 by design leads to more secure implementations. While we understand the need to allow the outdated legacy algorithm SHA-2 in early quantum-resistant standards like LMS, XMSS, and SLH-DSA, we think NIST should mandate the use of SHA-3 in all future standards.

John Preuß Mattsson,
Expert Cryptographic Algorithms and Security Protocols

# References

[1] Keeping Up with the KEMs: Stronger Security Notions for KEMs and Automated Analysis of KEM-based Protocols
https://eprint.iacr.org/2023/1933.pdf

[2] How to Hold KEMs
https://durumcrustulum.com/2024/02/24/how-to-hold-kems/

[3] Unbindable Kemmy Schmidt: ML-KEM is neither MAL-BIND-K-CT nor MAL-BIND-K-PK
https://eprint.iacr.org/2024/523.pdf

[4] Treating dishonest ciphertexts in post-quantum KEMs – explicit vs. implicit rejection in the FO transform
https://eprint.iacr.org/2025/062.pdf

[5] Anonymous, Robust Post-Quantum Public Key Encryption
https://eprint.iacr.org/2021/708.pdf

[6] On factoring integers, and computing discrete logarithms and orders, quantumly
http://kth.diva-portal.org/smash/get/diva2:1902626/FULLTEXT01.pdf

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

[8] lib25519 speed
https://lib25519.cr.yp.to/speed.html

[9] Signal documentation
https://signal.org/docs/

[10] Forward Secrecy for the Extensible Authentication Protocol Method for Authentication and Key Agreement (EAP-AKA' FS)
https://www.rfc-editor.org/rfc/rfc9678.html

[11] Enhancing Security in EAP-AKA' with Hybrid Post-Quantum Cryptography
https://datatracker.ietf.org/doc/html/draft-ar-emu-pqc-eapaka