ERICSSON

Date: April 1, 2025

Ericsson AB
Group Function Technology
SE-164 80 Stockholm
SWEDEN

Comments on NIST CSWP 39
Considerations for Achieving Cryptographic Agility

Dear NIST,

Thank you for your continued efforts in producing well-written, user-friendly, and open-access security documents. We appreciate the emphasis on cryptographic agility, as it plays a crucial role in strengthening security in practical systems.

However, the document's core message is somewhat difficult to extract. While cryptographic agility offers security benefits, it also introduces complexity, which is often considered a drawback in security design. For instance, the Suite B and CNSA suites minimize or eliminate cryptographic agility to reduce complexity. Additionally, many systems designed for crypto agility either never receive updates or must remain compatible with outdated peers, undermining its intended benefits. The growing trend of implementing crypto agility through software updates appears to be a healthy approach for software libraries. There are numerous cases where insecure in-band signaling, or user configuration has weakened security.

Please find below Ericsson's detailed comments on NIST CSWP 39:

- *"replacing a cryptographic algorithm in applications will require changes to application programming interfaces (APIs)"*

  We think this should be "may" or "can" instead of "will." Theoretically, you can switch between algorithms like AES-128, Camellia-128, and SM4 without modifying the software or hardware API. Similarly, migrating to X25519MLKEM768 in OpenSSL 3.5 typically does not require any API changes in the application. Applications using OpenSSL's default key exchange ("auto"), which should be the recommended approach, will after recompilation with the OpenSSL 3.5 library prefer X25519MLKEM768 [1]. No API changes are required.

- *"In a communication protocol, parties must agree on a common cipher suite, a set of cryptographic algorithms"*

  We think it would be better to remove "cipher suite" and instead refer to "a common set of cryptographic algorithms," as the term "cipher suite" typically imply fixed combinations.

- *"Since 2005, additional cryptanalyses have shown the weakness of SHA-1 with respect to not only collision resistance but also pre-image and second pre-image resistance [7]."*

  The cited paper only discusses collision attacks. To the best of our knowledge, no published pre-image attacks exist for full-round SHA-1.

- *"Each security protocol normally specifies a mandatory-to-implement algorithm"*

  This should be "mandatory-to-implement algorithms".

- *"For example, in TLS version 1.2 (TLSv1.2) [13] and TLS version 1.3 (TLSv1.3) [6], the version number specifies the default key derivation function, and the cipher suite identifier specifies the other algorithms."*

  "Some security protocols carry one identifier for each algorithm that is used, while other security protocols carry one identifier for a cipher suite that specifies the use of multiple algorithms. For example, in the IPsec protocol suite, Internet Key Exchange Protocol version 2 (IKEv2) [14] most commonly negotiates algorithms with a separate identifier for each algorithm. In contrast, TLSv1.3 [6] negotiates algorithms with cipher suite identifiers."

  The description of TLS 1.3 seems inaccurate. In TLS 1.3 [2], the Key Derivation Function (KDF) is determined by the selected cipher suite, there is no default KDF, except in the case of the PSK binder for external PSKs. In TLS 1.3, the AEAD and KDF algorithms are negotiated together using a cipher suite identifier, while the key exchange and signature algorithms are negotiated independently through extensions [2–3]. An example of a protocol that consistently uses a single cipher suite identifier to specify all cryptographic algorithms is EDHOC (RFC 9528) [4]. As two nodes compliant with NIST SP 800-52 will never negotiate TLS 1.2, we think the document should remove the reference to the outdated TLS 1.2.

- *"Security protocols include a mechanism to identify the algorithm or cipher suite in use. Some security protocols explicitly carry algorithm identifiers or a cipher suite identifier, while others rely on configuration settings to identify the algorithms or cipher suite."*

  "If a security protocol does not carry an explicit algorithm identifier, a new protocol version number or some other major change is needed to transition to a new algorithm or cipher suite."

  The text above appears to imply that crypto agility with protocol versioning is not a favorable approach. A new protocol version does not require a major change. We think the growing trend of implementing crypto agility through software patches rather than user configuration is a healthy approach for software libraries. As Filippo Valsorda explains in [5]: *"Today we acknowledge that promptly applying software patches for disclosed vulnerabilities is the cornerstone of secure software deployments, but ten years ago it was not standard practice. Changing configuration was seen as a much more rapid option to respond to vulnerable cipher suites, so the operator,*

through configuration, was put fully in charge of them. We now have the opposite problem: there are fully patched and updated servers that still behave weirdly, suboptimally, or insecurely, because their configurations haven't been touched in years."

Notably, WireGuard even lacks a version number. Unlike TLS, WireGuard operates with pre-established knowledge of the server's public key. This enables a straightforward and secure transition approach, binding each protocol version to a specific public key type. However, WireGuard's plans for migrating to new cryptographic algorithms remains unclear. WireGuard's approach seems to be similar to Suite B, CNSA 1.0, and CNSA 2.0, which also leave migration between profile versions unspecified. We believe that it is important the the security protocol specifies how to securely transition to new algorithms and cipher suites. Relying on applications to handle protocol version negotiation increases the risk of vulnerabilities.

- We recommend that NIST require all future algorithms to be fully specified, including domain parameters and all other parameters such as key length, hash function, etc. Both symmetric and asymmetric keys and algorithms should be bound to a fully specified algorithm identifier and used exclusively with that algorithm. Exceptions should be permitted only when the use of different algorithms or parameters has been proven secure [6]. One such secure approach involves using a seed as the private key, with key derivation to generate expanded private keys for different algorithms, such as ML-KEM and ML-DSA.

- Since a cryptographic key is always associated with a set of algorithms (e.g., AES, RSA, ECC) and should be explicitly bound to a specific one (e.g., ECDSA with P-384 and SHA-384, following FIPS 186-5), key expiration and revocation are important tools for cryptographic transitions and should be discussed in the document.

- In the section "Providing Notices of Expected Changes," we believe it is important to discuss that publicly available information does not provide the full picture. Government agencies conduct their own cryptanalytic research, which is typically not disclosed, meaning that algorithms may be significantly more vulnerable than publicly known. Historical examples include Alan Turing's cryptanalysis of Enigma, SIS's cryptanalysis of Purple, NSA's early knowledge of differential cryptanalysis, and Arne Beurling's decryption of the Geheimschreiber, an effort that, along with "apps" developed by Ericsson, enabled the interception of Nazi communications passing through Sweden [7–10].

  We believe NIST should establish transition timelines under the assumption that the actual security strength of algorithms may be lower than publicly available information suggests.

- *"Cryptographic algorithm selection or negotiation should have its integrity protected"*

  The document should also discuss that, in general, security protocols should support in-band negotiation of cryptographic algorithms. An exception to this is a system where a server can securely inform clients out-of-band about the only cryptographic configuration they should use. In-band negotiation can also occur at a higher level, such as negotiating a protocol version or a suite of algorithms (e.g., EDHOC) rather than individual algorithms. This aligns with the broader trend of fixing hash functions in standardized digital signature schemes such as Ed448 and ML-DSA. Additionally, highly trusted algorithms like AES, SHA-2, and SHA-3 have demonstrated

remarkable stability over the past decades, reducing the necessity of frequent algorithm negotiation. From a security perspective, negotiating individual algorithms may not always be necessary, as a sudden and severe break of a widely trusted algorithm would be an extraordinary event. However, fine-grained algorithm negotiation can still be practical in certain scenarios.

Additionally, it should emphasize the critical distinction between systems that enable secure negotiation of cryptographic algorithms (e.g., TLS 1.3) and those that do not. In systems with secure negotiation, simply introducing a stronger algorithm enhances security, if both parties prefer the new algorithm, they will never negotiate a weaker, outdated one. In contrast, systems without secure algorithm negotiation only improve security when weaker algorithms are explicitly disabled. This is because security in such systems is determined by the weakest integrity algorithm that the receiver is willing to accept. For systems lacking secure algorithm negotiation, it is even more crucial to rapidly support newly standardized algorithms. Early adoption of stronger algorithms facilitates their widespread use, ensuring that weaker algorithms can eventually be phased out without disrupting interoperability.

- *"In most cases, choosing a hybrid algorithm leads to a second transition when the traditional algorithm is deprecated"*

  "However, the cost of deploying a PKI root of trust is significant, so the expense associated with a transition to the use of a hybrid root of trust followed by a second transition to using only a PQC algorithm for a root of trust must be considered."

  "That is, these people will continue to use the hybrid algorithm even when the traditional algorithm is no longer secure."

  If high-performance hybrids like X-Wing or X25519MLKEM768 are deployed in the first transition, the necessity of a second transition becomes highly uncertain. Since NIST permits the use of disallowed algorithms within hybrid schemes [11], any second transition will be entirely unrelated to the deprecation of traditional algorithms. The potential performance gains are unlikely to justify a second transition of existing systems, but standalone ML-KEM may be considered for new systems. If a CRQC is ever built, early CRQCs will most likely be very expensive, meaning that early attackers will focus on very high-value targets [12]. Even disallowed algorithms can significantly increase the practical cost of otherwise trivial attacks on weak PQC implementations.

- *"In addition, the use of hybrid key-establishment algorithms increases bandwidth usage because more data needs to be exchanged, which can be a problem for some implementation environments."*

  The ML-KEM-768 encapsulation key is 1,184 bytes, and the ML-KEM-768 ciphertext is 1,088 bytes, whereas X25519 public keys are only 32 bytes. The increase from X25519 to ML-KEM-768 is a staggering 3450%, while the jump from ML-KEM to state-of-the-art hybrid algorithms like X-Wing or X25519MLKEM768 is approximately 3%. It is hard to argue that the modest increase from a non-hybrid to a hybrid PQC scheme is a significant concern in the face of the massive changes introduced by either scheme. We suggest rewriting to state that the bandwidth increase from hybrids is not a problem and instead discussing the challenges of deploying standalone ML-KEM in very constrained radio networks, which is often not possible [13].

- The "Cryptographic Key Establishment" section focuses on whether the security analysis of a protocol is straightforward or difficult. However, many completely insecure protocols are easy to analyze, so this perspective gives a way too negative impression of EAP. In reality, using EAP-TLS [14] exclusively, without allowing other EAP methods, is common and typically far more secure than designing a new key establishment mechanism from scratch. Additionally, EAP offers a flexible framework for rapidly transitioning to more secure key establishment methods. For instance, while public 5G networks rely on AKA without asymmetric key exchange, private 5G networks can already leverage EAP-AKA'-FS with X25519 and X-Wing (ML-KEM-768 + X25519 + SHA3-256) for significantly stronger security, aligning with best practices [15–16]. We think NIST should recommend always hybridizing symmetric keying with post-quantum secure asymmetric keying when possible [17]. Additionally, we suggest that the document highlight EAP-AKA'-FS with X25519 and X-Wing as examples on how EAP can enable cryptographic agility and facilitate migration to PQC.

- *"When specifying a cipher suite, the relative strength of each algorithm should be roughly equal. Complexity in security protocols needs to be avoided."*

  If the performance of a particular type of algorithm does not impact overall performance, specifying its highest strength across all cipher suites can reduce complexity.

- *"For example, in the case of IPsec, the datagram encryption and authentication provided by IPsec need to operate in the kernel."*

  Not when using RFC 3948, "UDP Encapsulation of IPsec ESP Packets." We suggest that the sentence is updated to clarify this. It may also be worth mentioning that ESP operates in kernel space, whereas IKEv2 runs in user space. The term 'IPsec' is often used to refer to both IKEv2 and ESP together.

- *"In some operating systems, only a subset of the crypto API's overall capabilities is available in the kernel. This subset is determined by the cryptographic operations required in the kernel. In many operating systems, the supported algorithms in the kernel are established when the kernel is built, meaning that plugins to add algorithms are not available in the kernel."*

  We appreciate that NIST discusses cryptographic APIs between user space and kernel space. 3GPP made a critical mistake by specifying the use of DTLS over SCTP for 5G interfaces, under the mistaken belief that this was a recommended IETF security standard where confidentiality, integrity, and replay protection are all provided by DTLS. In reality, integrity and replay protection are handled by SCTP-AUTH, an outdated and insecure protocol developed outside of IETF's security area. Efforts to standardize a secure replacement have faced significant delays due to demands for an API between kernel and user space. Such an API introduces major challenges, as it prevents the use of hardened and certified DTLS libraries without modifications. Altering DTLS libraries not only increases the risk of security vulnerabilities but also complicates the rapid deployment of critical security patches.

- *"Some chips, like Subscriber Identity Module (SIM) cards and TPMs, are dedicated to cryptographic operations. These chips are part of a larger computer system like a mobile phone or a laptop computer. These chips store the private keys and perform cryptographic operations that depend on the keys. At no time does the private keying material leave the chip. These chips*

support very few cryptographic algorithms, and changing algorithms is accomplished by replacing the chip."

The USIM application resides on a removable UICC or a non-removable eUICC or iUICC. Modern UICCs and TPMs support firmware updates, while UICCs also allow OTT application updates. eUICCs allows over-the-top (OTT) upgrades of the entire USIM application at any time, including support for new cryptographic algorithms. The GSMA IoT SAFE standard also allows the eUICC to function as a TPM. This TPM functionality can reside directly within the eUICC or be implemented as part of the USIM application. IMSI encryption may be handled within the USIM but can also be performed on the mobile device. Additionally, USIM algorithms for authentication, key generation, and IMSI encryption support private code points, enabling the use of algorithm profiles such as CNSA without requiring standardization. Unlike removable UICCs, which can be replaced, non-removable UICCs and TPMs are difficult to replace. Additionally, the keying material on removable UICCs is provisioned during manufacturing, making it accessible outside the chip, an exposure that attackers have exploited, see for example [18]. In line with zero-trust principles, it is crucial to always assume compromise, and to implement mechanisms such as [14−15] to mitigate the impact of compromises.

- *"For example, the Intel SHA Extensions paper [27] states that the CPUs offer features to make SHA hash computations faster."*

  Instead of solely referencing a paper on SHA-1 and SHA-256 acceleration, we think the document should highlight more forward-looking CPU architectures, such as those from ARM, IBM, and RISC-V, that include features designed to optimize SHA-3 performance. The SHA-3 family is required by ML-KEM, ML-DSA, and Ed448 and plays a crucial role in the transition to PQC.

- *"Current practice has made it possible for applications to access cryptographic services through APIs. This significantly eases a cryptographic transition from one algorithm to another."*

  The document should also mention that APIs often significantly complicates and delays cryptographic transitions from one algorithm to another. Here are a few examples:
  - The fixed-length output of SHA-2 complicates transition to SHAKE256.
  - The Extract-Expand API of HKDF complicates transition to KMAC.
  - Not supporting the AES round function complicates transition to AEGIS and Rijndael-256.
  - Assumption on block size complicates transition to Rijndael-256.
  - Not supporting Keccak-p complicates the adoption of TurboSHAKE.
  - Weierstraß-based APIs complicated the transition to Curve25519 and Curve448.
  - API assumptions on small key lengths, complicated transition from RSA-1024 and -2048.
  - API assumptions on a small, fixed signature size complicate transition to PQC signatures.
  - APIs designed around PBKDF2 complicate transition to Argon2.
  - APIs designed for independent encryption and integrity complicated transition to AEAD.

John Preuß Mattsson,
Expert Cryptographic Algorithms and Security Protocols, Ericsson

References

[1] OpenSSL 3.5.0
https://github.com/openssl/openssl/releases/tag/openssl-3.5.0-alpha1

[2] RFC 8446, The Transport Layer Security (TLS) Protocol Version 1.3
https://www.rfc-editor.org/rfc/rfc8446.html

[3] The Illustrated TLS 1.3 Connection
https://tls13.xargs.org/

[4] RFC 9528, Ephemeral Diffie-Hellman Over COSE (EDHOC)
https://www.rfc-editor.org/rfc/rfc9528.html

[5] Automatic Cipher Suite Ordering in crypto/tls
https://words.filippo.io/dispatches/cipher-suite-ordering/

[6] On using the same key pair for Ed25519 and an X25519 based KEM
https://eprint.iacr.org/2021/509

[7] Arne Beurling
https://en.wikipedia.org/wiki/Arne_Beurling

[8] Decyphering the Geheimschreiber, a Machine Learning approach
https://www.diva-portal.org/smash/get/diva2:1337816/FULLTEXT01.pdf

[9] The Breaking of Geheimschreiber
https://www.nsa.gov/portals/75/documents/news-features/declassified-documents/crypto-almanac-50th/The_Breaking_of_Geheimschreiber.pdf

[10] FRA:s historia
https://fra.se/omfra/frashistoria.4.55af049f184e92956c42c71.html

[11] NIST Comment on the Transition to Post-Quantum Cryptography Standards
https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/uHMw8RNGkC8/m/AjupvtbgBwAJ

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

[13] Constrained Radio Networks, Small Ciphertexts, Signatures, and Non-Interactive Key Exchange
https://csrc.nist.gov/csrc/media/Events/2022/fourth-pqc-standardization-conference/documents/papers/constrained-radio-networks-pqc2022.pdf

[14] RFC 9190, EAP-TLS 1.3: Using the Extensible Authentication Protocol with TLS 1.3
https://www.rfc-editor.org/rfc/rfc9190.html

[15] RFC 9678, Forward Secrecy Extension to the Improved Extensible Authentication Protocol Method for Authentication and Key Agreement (EAP-AKA' FS)
https://datatracker.ietf.org/doc/rfc9678/

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

[17] 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

[18] The Great SIM Heist
https://theintercept.com/2015/02/19/great-sim-heist/