Public Comments on the SP 800-56 Subseries

Comment period: December 11, 2024 - January 31, 2025

On December 11, 2024, NIST's Crypto Publication Review Board initiated a review of the SP 800-56 series of NIST publications.

The comments that NIST received during the comment period are collected below.

More information about this review is available from NIST's Crypto Publication Review Project site.

LIST OF COMMENTS

1. Comments from Kaleb Himes (wolfCrypt), 12/11/2024

Rationale for Including X25519 and X448 in NIST SP800-56A Updates

- Modern Cryptographic Standards:
  - X25519 and X448 are key exchange protocols designed to meet the demands of modern cryptography, offering both strong security and efficiency
  - Their inclusion aligns with NIST's goal of standardizing robust and efficient cryptographic algorithms
- Wide Adoption and Proven Security:
  - X25519 and X448 are widely adopted in industry standards such as TLS 1.3, SSH, and Signal
  - They have undergone extensive analysis and have proven to be secure against known cryptanalytic attacks, making them a trusted choice for secure communications
- Efficient Implementation and Performance:
  - Both algorithms provide high-performance implementations suitable for a range of platforms, from constrained IoT devices to high-throughput servers
  - They are specifically designed to minimize implementation complexity, **reducing the risk of side-channel attacks**
- Interoperability and Ecosystem Support:
  - Including X25519 and X448 in SP800-56A ensures better alignment with global cryptographic standards (e.g., RFC 7748) and facilitates interoperability between U.S. standards and international frameworks
  - This strengthens the cryptographic ecosystem and enhances global adoption of secure protocols
- Long-Term Security and Post-Quantum Considerations:
  - X25519 and X448 are designed to provide long-term security, with X448 offering a higher security level for future-proofing against advances in computational power
  - While not post-quantum algorithms, their robust security complements the transition to hybrid or post-quantum schemes.
- Alignment with Modern Curve Standards:
  - These protocols use modern elliptic curves (Curve25519 and Curve448) that are specifically chosen for their simplicity and resistance to implementation pitfalls, avoiding the potential issues present in legacy curves.

Please bring back X25519 and X448 like we had under FIPS 140-2 with IG A.2 but this time as “NIST Recommended Curves”!

2. Comments from Mark Wagner, 12/16/2024

The default_salt and "Suggested Maximum Byte Length of salt" for Single-Step KDF with H(x) = KMAC# should be 5 bytes (not 4) smaller than a single block of input to KMAC.

Context:
1) Section 4.1 Specification of Key-Derivation Functions, Implementation-Dependent Parameters #3 default_salt, states that:
  a) the default_salt for H(x) = KMAC128 shall be an all-zero string of 164 bytes, and
  b) the default_salt for H(x) = KMAC256 shall be an all-zero string of 132 bytes.

2) Section 4.2 The Auxiliary Function H(x) and Related Parameters, Table 3: H(x) = KMAC#, column "Suggested Maximum Byte Length of salt" specifies 164 and 132 bytes for KMAC128 and KMAC256, respectively.
The table shows that both of these sizes are 4 bytes smaller than the input block size of their respective KMAC.
The footnotes explain that the 4 bytes are needed to leave room for the encoded length of the salt.
The stated goal is that the encoded salt is no more than 1 block of input to KMAC.

Issue: I believe that the encoded length of the salt is 5 bytes long, not 4.

Consider KMAC128. From NIST SP 800-185 SHA-3 Derived Function, section 4.3 KMAC Definition, we have:

encoded salt = bytepad(encode_string(salt), 168)
    = bytepad(left_encode(len(salt)) || salt, 168)
    = left_encode(168) || left_encode(len(salt)) || salt || zero pad to even 168-byte blocks

For a max-sized/default salt of 164 bytes, we have len(salt) = 1312 bits (note that len(X) is in bits), so that

encoded salt = (1, 168) || (2, 0x05, 0x20) || 164-bytes salt || zero pad   (where 0x0520 = 1312)
    = 169 bytes of encoded salt + 167 bytes of zeros
    = 2 blocks of 168-bytes each

Clearly, we are off by 1 byte of the stated goal of 1 block of input to KMAC.

A possible explanation is that SP 800-56Cr2 thought that len(X) was in bytes, not bits, so that the prefix to salt would be (1,168) || (1,164).

KMAC256 is similar with a prefix of (1, 136) || (2, 0x04, 0x20), where 0x0420 = 1056.

Fix: Update section 4.1 default_salt and section 4.2 Table 3 (and footnotes) to reflect max salt sizes of 163 and 131 bytes for KMAC128 and KMAC256, respectively; 5 bytes less than their input block sizes.

Best,
Mark Wagner

3. Comments from Dimitri Ledkov (Chainguard Inc.), 12/17/2024

Please standardise X25519 and X25519MLKEM768 as approved KEM.

Please collaborate with IETF for simultaneous approval and compatibility with https://datatracker.ietf.org/doc/draft-ietf-tls-hybrid-design/

Please reopen FIPS 140-2 module updates to add X25519 support.

These actions will significantly simplify and aid adoption of NIST approved algorithms in all contexts.

It will also accelerate RSA/ECDSA depreciation to attempt PQC-only by 2035 timeframe.

4. Comments from John Mattsson (Ericsson), 1/23/2025

Dear NIST,

Thanks for your continuous efforts to produce well-written, user-friendly, and open-access security documents. Thanks for your continuous efforts to produce open-access security documents. Please find below our comments on NIST SP 800-56Ar3, 800-56Br2, and 800-56Cr2:

General comments applicable to multiple specifications:

- [ISO/IEC 8825] is a paywalled standard that provides no additional value, as DER encoding is fully specified in the open-access ITU-T X.690 [1]. We believe that security standards hidden behind paywalls are unacceptable [2]. Paywalls significantly discourage security researchers from analyzing and reasoning about standards and their interactions. Open access is critical for security specifications, as history has repeatedly demonstrated that a lack of open discussion and thorough analysis often results in significant vulnerabilities. Paywalled standards also enable selective distribution of weakened standards to specific users, similar to the Swiss Crypto AG affair [3]. Paywalled standards pose a significant cybersecurity risk.

  We strongly recommend that NIST remove all references to paywalled standards and replace ISO/IEC 8825 with the freely accessible ITU-T X.690 [1]. This approach aligns with essential Internet Engineering Task Force (IETF) standards, including RFC 2986, 4210, 4511, 5280, 5652, 6960, 7906, and others, which consistently reference [1].

- The three specifications provide detailed definitions, including security properties, for hash functions, MACs, and PRFs. However, a similar definition for Key Derivation Functions (KDFs) is missing. We recommend that NIST define a KDF as a PRF that maintains collision resistance and preimage resistance even when the key is known. Both KMAC and HMAC satisfy these requirements and should be recommended as general-purpose KDFs.

- Key encapsulation should be included in the definitions, introduction, and other relevant sections.

- The security strength should be aligned with the five new quantum-resistant security categories.

- Deprecated algorithms, such as SHA-1, should be removed.

Comments on 800-56A:

- We strongly support the 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 exploitable vulnerabilities. These flaws have allowed attackers to recover the so-called “ephemeral” private key, enabling them to completely compromise sessions between legitimate parties. We strongly recommend that the requirement “An ephemeral private key shall be used in exactly one key-establishment transaction” be explicitly included in all NIST specifications on Key Encapsulation Mechanisms (KEMs). Any protocol that reuses private keys should explicitly acknowledge this practice and state that the keys are (semi-)static.

- We think NIST should spend limited time updating 800-56A, as it includes algorithms that are expected to be phased out soon. While X25519 and X448 are critical for hybridizing ML-KEM and should be added to 800-56A, we see no future need for FFDH and Weierstraß curves. Montgomery curves offer superior security and performance and should be the sole choice for ML-KEM hybridization. Hybrids with X25519 have already become the de facto standard for TLS 1.3, DTLS 1.3, QUIC, and SSH. We think NIST should encourage the use of X25519 and X448, as specified in SP 800-186, in hybrid schemes.

  As a less favorable alternative to adding X25519 and X448 to 800-56A, NIST could promote the use of X25519 and X448 as non-approved auxiliary functions in combination with ML-KEM in hybrid schemes. This minor adjustment would still support the adoption of PQC and align with de facto Internet standards.

- We don’t think [ANS X9.42] and [ANS X9.63] are to be considered normative references, and we recommend moving them to the informative references section. If any part of [ANS X9.42] or [ANS X9.63] is deemed normative, NIST should revise 800-56A to make sure that [ANS X9.42] and [ANS X9.63] are no longer normative. Paywalled security standards constitute a cybersecurity risk, and we believe NIST should remove all normative references to such standards.

- “Advanced Encryption Standard (AES) as defined in Federal Information Processing Standard (FIPS) 197,1 and the Keyed-Hash Message Authentication

Code (HMAC) as defined in FIPS 1982 make attractive choices for the provision of these services."

Given the narrow block length of AES, the fact that SHA-2 does not behave like a random oracle, and SHA-2's vulnerability to length-extension attacks, we believe that FIPS 202, SP 800-185, and SP 800-197 are more attractive choices.

Comments on 800-56B:

- We do not think NIST should spend time on updating or revising 800-56B. Both Ericsson and 3GPP agreed with NSA Suite B that integer factorization cryptography should be phased out. While X25519, X448, Ed25519, and Ed448 will be crucial for hybridizing ML-KEM and ML-DSA, we see no future need for integer factorization-based cryptography.
-
- [ANS X9.44] is not a normative reference and should be moved to informative references.

Comments on 800-56C:

- We welcome NIST's plans [4] to allow hybrid shared secrets of the form Z' = T || Z. However, since many systems are expected to use hybrid shared secrets consisting of more than two components, we recommend that NIST extend support to hybrid shared secrets of the form Z' = T_1 || Z || T_2, where T_1 and T_2 auxiliary shared secrets. This structure, Z' = T_1 || Z || T_2 should be secure for all KDFs with acceptable security properties. We anticipate that hybrid shared secrets consisting of more than two components will be relatively common. For instance, [5], 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, ML-KEM, and BIKE/HQC has been proposed as an alternative to FrodoKEM with better performance.

  We recommend that NIST impose reasonable requirements, such as length limits on T_1 and T_2 . There is no justification for allowing megabyte-sized T. It is good that 800-56C already mandates that "the method used to generate T must be known and agreed upon by all parties." The suggestion in [6] to allow hybrid shared secrets of the form $S_1 \| S_2 \| \ldots \| S_t$, where at least one of the shared secrets $S$$ is NIST approved seems perfect.
- *"When an approved MAC algorithm (HMAC or KMAC)"*

We suggest revising this to "When an approved hash-based PRF (HMAC or KMAC)," as other approved MAC algorithms, such as CMAC and GMAC, do not meet the required properties [7].

- We recommend that NIST restrict the use of SHA-2 and AES-CMAC as KDFs. Both are unsuitable in scenarios where one of the parties can control the input. SHA-256 and SHA-512 are vulnerable to length-extension attacks, while the issues with AES-CMAC are detailed in [7]. Although SP 800-108 [8] proposes a potential solution for AES-CMAC, it appears ad hoc, lacking both motivation and proof. We believe SHA-2 should be explicitly disallowed as a KDF and removed from the specification. Systems relying on SHA-2 can and should use HMAC. Additionally, we emphasize that NIST should discuss length-extension attacks in FIPS 180-5, as suggested in [9].
  For AES-CMAC, we argue that it should not be permitted as a KDF unless it can be demonstrated that a specific construction and input formatting ensure collision resistance and preimage resistance, even when the key is known. A potential solution could be moving AES-CMAC to an appendix and limiting its use to legacy systems. We note that truncated CMAC is an excellent MAC, which behaves like an ideal MAC, even against multiple forgery attacks.

- We think the specification should talk less about MAC algorithms and instead talk about PRFs. While all PRFs can serve as MACs, a MAC generally cannot serve as a PRF.

- In Section 7.1, “Selecting Hash Functions and MAC Algorithms,” we think the recommendation should be to use KMAC whenever possible. AES and SHA-2 have significant problems with sidechannels, AES-CMAC and SHA-2 are unsuitable as a KDF when one party can control the input [7], and KMAC offers stronger security properties compared to HMAC. We applaud NIST for mandating the use of SHA-3 in ML-KEM and ML-DSA.

- While Section 8.1, “Using a Truncated Hash Function,” correctly notes that truncation may be less efficient than using the corresponding untruncated version, truncation significantly enhances the security properties of algorithms such as SHA-512, CMAC, and HMAC. For example, untruncated SHA-512 provides no protection against length-extension attacks, whereas SHA-384 offers 128- bit security against such attacks. Similarly, untruncated CMAC and HMAC deviate significantly from the behavior of ideal MACs, whereas their truncated counterparts do behave as ideal MACs when the number of queries is limited. A general recommendation should be to use SHA-2, HMAC, and CMAC with a longer-than-required output size and truncate the result. We believe Section 8.1

should either be removed or rewritten to highlight the positive security properties of truncation. Ericsson has historically aligned with NSA Suite B and CNSA 1.0, using (HMAC-)SHA-384 wherever possible. During the PQC migration, we plan to adopt the superior SHA-3 family as widely as possible.

John Preuß Mattsson,

Expert Cryptographic Algorithms and Security Protocols

1. ITU-T X.690, ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER) https://www.itu.int/rec/T-REC-X.690/

2. Ericsson Comments on SP 800-38E https://csrc.nist.gov/csrc/media/Projects/crypto-publication-review-project/documents/decisionproposal-comments/sp800-38e-decision-proposal-comments-2023.pdf

[3] Crypto AG https://en.wikipedia.org/wiki/Crypto_AG

[4] Ordering of Shared Secrets in SP 800-56C Combiner https://groups.google.com/a/list.nist.gov/g/pqc-forum/c/ST_yMzYyMl0/m/hEMfXBFHCgAJ

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

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

[7] Ericsson Comments on SP 800-38B https://csrc.nist.gov/csrc/media/Projects/crypto-publication-review-project/documents/initialcomments/sp800-38b-initial-public-comments-2024.pdf

[8] NIST SP 800-108, Recommendation for Key Derivation Using Pseudorandom Functions https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-108r1-upd1.pdf

[9] Ericsson Comments on FIPS 180-4 https://csrc.nist.gov/csrc/media/Projects/crypto-publication-review-project/documents/initialcomments/fips180-4-initial-public-comments-2022.pdf

5. Comments from Julien Eynard (Rambus Inc.), 1/29/2025

Thank you for your work in publishing and maintaining these standards documents. Please find below our comments on NIST SP 800-56.

Allow x-only elliptic-curve Diffie-Hellman key exchange (SP800-56A)

We recommend that NIST update SP 800-56A to allow x-only elliptic-curve Diffie-Hellman key exchange (called ECC CDH in SP 800-56A) for curves modulo an odd prime p. The ECC CDH primitive specified in §5.7.1.2 outputs only the x-coordinate of the agreed point P. This coordinate depends on the x-coordinate of the input point Q, but not on its y-coordinate. Encoding only the x-coordinate reduces network traffic, and in our experience this optimization is used already in industry. However, it is important to validate that the given x-coordinate really corresponds to a point on the elliptic curve.

Standardizing x-only ECC CDH would require a few changes. We suggest that an input $x_Q \in [0, p-1]$ should be given, and used to recover a point $Q' \coloneqq (x_Q, y_Q')$ on the elliptic curve, where $y_Q' \in [0, p-1]$ and $\left(y_Q'\right)^2 = x_Q^3 + ax_Q + b$. This point Q' will equal either the original public key Q or its additive inverse. Then Q' can be used with the existing ECC CDH primitive, including validation as appropriate. The resulting shared secret x_P will be the same when using Q' as it would have been with Q.

As with other cryptographic routines, the specification should allow mathematically equivalent implementations, such as checking that $y_Q' \in \mathbb{F}_p$ using the Legendre symbol, and then computing the x-coordinate of the shared secret point hdQ' without recovering y_Q' — for example, using the Brier-Joye ladder.

Allow X25519 and X448 (SP800-56A)

Similar to x-only ECC CDH, we recommend that NIST allow the use of X25519 and X448 key exchange, as specified in RFC 7748. These key exchange protocols are simpler to implement than the ones using short Weierstrass curves, as well as having the lower communication requirement of x-only ECC CDH. The protocols are broadly deployed, they have high-quality open-source implementations, and FIPS 186-5 already allows the isogenous curves Ed25519 and Ed448 for signatures.

We recommend that, per RFC 7748, cryptographic modules are allowed to check that an X25519 or X448 public key really represents a point on the curve (rather than the twist), but that they are not required to do so. Omitting this check is considered safe because X25519 and X448, unlike the standard short Weierstrass curves, have 4·prime order twists. While the check itself is straightforward, omitting it facilitates code reuse.

Allow deterministic key derivation (SP800-56A)

We recommend that NIST allow deterministic generation of asymmetric keys from a fixed seed. In particular, it should support having a module-unique random secret (e.g., placed in NVM at manufacturing/provisioning or generated by a physically unclonable function) and using a standard key derivation function to compute the fixed seed.

The first advantage of this is that context can be provided as additional input in the key derivation function allowing the derivation of multiple keypairs bound to the module. The second advantage is that the module does not have to store the expanded public / private parts of the keypair(s). The long-term secret could be considered a root key-derivation key (KDK) from which could be derived a seed to re-generate said keypair, according to the two possible ECC key generation techniques described in SP800-56A paragraphs 5.6.1.2.1 and 5.6.1.2.2 for example.

We believe that standardizing and allowing such mechanisms would be highly beneficial as there are ongoing discussions without clear guidance about this topic (see e.g., attestation services).

Beyond the SP800-56 series, digital signature keypairs (FIPS 186-5, and FIPS 204 ML-DSA), as well as key encapsulation mechanisms (FIPS 203 ML-KEM) could benefit from such approved deterministic asymmetric key derivation. A direct example for digital signatures is DICE-like attestation services where the DeviceID keypair is deterministically derived from components attesting the module identity. We recommend that digital signature standards and SP800-56 get aligned to support deterministic key generation as required for DICE.

We are aware of the guidance note “Allowed Asymmetric Key Derivation Methods” from July 24th, 2023 that NIST has provided to the CMUF group, but we recommend to go a step further and officially add this to SP800-56.

Allow for hybrid and pure PQC key derivation in line with IETF (SP800-56C)

We recommend that NIST allows both hybrid and “pure PQC” key derivation approaches and we see an urgent need to have both approaches standardized. There are good proposals for key derivation from hybrid and “pure” sources being discussed at IETF. We do not want to rehash the arguments from IETF mailing lists – we know that NIST is aware of the discussions and participates in them. Neither do we want to comment on any specific proposal. However, we want to state explicitly that we do support both hybrid and “pure” approaches. We also want to emphasize the need to align between NIST and IETF to ensure that standardized solutions from IETF protocols are FIPS 140-3 certifiable. Otherwise chip and device vendors will see significant delays in transitioning their products to quantum secure, FIPS 140-3 certifiable hardware and firmware.

Avoid normative references that are not openly available (SP800-56)

We notice that SP800-56Br2 includes a normative reference to ISO/IEC 8825 which is only available behind an ISO paywall. We would like NIST to consider not using normative references that are paywalled.

Allow for plenty of discussion and feedback (SP800-56)

We recommend NIST to allow more time for discussion and feedback than usual. Updating SP800-56 will be a significant effort. In particular, we expect that an initial public draft for public review will not be sufficient before the final documents are published. At a minimum, a second draft for public review should be planned for. A dedicated mailing list for public discussion of the drafts may also be useful.

Kind regards,

Julien Eynard
Mike Hamburg
Simon Hoerder
Stuart Kincaid

On behalf of the Rambus security certification team

6. Comments from Julien Eynard (Rambus Inc.), 1/31/2025

We would like to you to consider the following addendum to our comments:

“We recommend that NIST clarify in SP800-56A §5.6.2.3.3 and 5.6.2.3.4 (ECC public key validation) that mathematically redundant tests need not be performed. For example, if a point Q given by affine coordinates (xQ,yQ) satisfies the curve equation of an elliptic curve of prime order n, then nQ = identity need not be checked because it holds for all points on the curve. Likewise, Q != identity need not be checked because the identity cannot be written in affine coordinates.”

7. Comments from Swapneela Unkule (atsec), 1/31/2025

#	Line #	Comment (Include rationale for comment)	Suggested change. (if applicable)
1	SP 800-56Ar3 Section 5.7.1.2	The ECC CDH primitive as defined in Section 5.7.1.2 uses only the x coordinate of P to create the shared secret Z. Because this approach is very common in practice (both academia and industry), researchers have created formulas ("ladders") to compute only the x coordinate of P, given the x coordinate of QB and dA (and h if applicable). One such example is https://eprint.iacr.org/2020/437.pdf. While this is not mathematically equivalent to computing step 1 and 2 individually (as information about the y coordinate is lost), it is mathematically equivalent to computing step 1, 2, and 3 taken together. It would be beneficial if NIST could explicitly allow this approach in the standard. However, this does introduce some subtle points regarding the assurances: • For ECC partial public key validation (5.6.2.3.4). Step 1 would not be applicable as an x coordinate on its own cannot represent the identity element. Step 2 would omit the check on the y coordinate as it is not present. Step 3 would be omitted completely as the y coordinate is not present, so any x coordinate within the range checked by step 2 could be valid. • For ECC full public key validation (5.6.2.3.3). Step 1-3	Explicitly allow ECC CDH with only x coordinates.

		are addressed above. Step 4 may be impossible to address. For prime order curves, this step will always be met, but for binary or Koblitz curves the x coordinate could correspond to a point in a smaller subgroup. Perhaps full public key validation cannot be performed in this scenario.	
2	SP 800-56Ar3 Section 5.6.2.1.4	Bullet a) in this section states that the key pair owner can obtain assurance of pair-wise consistency by generating the key pair themself, following Section 5.6.1. However, FIPS 140-3 IG 10.3.A, additional comment 1, requires that a FIPS module performs a pair-wise consistency test after generating a key pair: "At minimum, the PCT that is required by the underlying algorithm standard (e.g., SP 800-56Arev3 Section 5.6.2.1.4 option 'b' or SP 800-56Brev2 Section 6.4.1.1 option '2.') shall be performed, which will satisfy the PCT requirement in either TE10.35.01 for key transport, TE10.35.02 for signatures, or TE10.35.03 for key agreement." Evidently, bullet a) in this section is not applicable in the context of FIPS modules, and bullet b) introduces a "redundant" operation, where the module simply recomputes the public key twice. Nevertheless, bullet b) is commonly implemented by FIPS modules to satisfy the IG 10.3.A requirement, regardless of whether static or ephemeral key pairs are generated (note that SP	Update bullet b as follows: "For either a static key pair or an ephemeral key pair: Subsequent to the generation ..." Add the following bullet: c) For either a static key pair or an ephemeral key pair: Subsequent to the generation of a key pair by the owner or a trusted third party as specified in Section 5.6.1, the owner performs one of the following consistency tests (as appropriate for the FFC or ECC domain parameters used during the generation process). • For an FFC key pair (x, y): a known good key pair (s, t) must be available. Use the private key, x, and the public key, y, along with the prime modulus p included in the domain parameters associated with the key pair, to compute tx mod p and ys mod p. Compare the results. If the results are not equal, then the pair-wise consistency test fails. • For an ECC key pair (d, Q): a known good key pair (e, R) must be available. Use the

		800-56Ar3 allows b) only to be used for static key pairs). To alleviate this issue, we suggest that: • bullet b) is allowed for ephemeral key pairs; this is valid as b) already requires these steps to be performed after generation. • a third method to obtain assurance of pair-wise consistency is added: generating a shared secret with a known good key pair. We strongly recommend that NIST CT group confers with the CMVP regarding this proposal.	private key, d, and the public key, Q, along with the domain parameters associated with the key pair, to compute dR and eQ (according to the rules of ellipticcurve arithmetic). Compare the results. If the results are not equal, then the pair-wise consistency test fails.
3	SP 800-56Ar3 Section 5.6.2.3.3	Step 4 of the process described in Section 5.6.2.3.3 can almost always be simplified. For instance, Section 3.2.2.1 of SEC 1 Version 2.0 (May 21, 2009) brings a similar set of requirements as to those of SP800-56Ar3, and the former document states a simplification method: "In Step 4, it may not be necessary to compute the point nQ. For example, if h = 1, then nQ = O is implied by the checks in Steps 2 and 3, because this property holds for all points Q ∊ E. If h = 2 and T represents elliptic curve domain parameters over F2m , then it suffices to check that the trace of xQ is 1. Similar checks may be performed in other situations where h is small." SP 800-56Ar3 currently approves only curves over GF(p) (i.e., h = 1) and GF(2m). SP 800-56Ar3 currently approves only curves over GF(p) (i.e., h = 1) and GF(2m). Therefore, this step could	Add the following text to Step 4 (below the paragraph "Success at this stage ensures that the public key ... and is not the identity element."): If h = 1, this step may be omitted. If h = 2 and q = 2m, this step may be replaced by verifying that the trace of xQ is 1.

		be omitted (for prime order curves) or simplified (for binary and Koblitz curves) in the same manner as SEC 1 specifies.	
4	SP 800-56Ar3 5.6.2.2.3.1	Recipient's assurance from TTP is listed as "The methods used by a third party trusted by the recipient to obtain that assurance are beyond the scope of this Recommendation"	It would be helpful to provide some guidance to the vendor on the methods to be used.

8. Comments from Kris Kwiatkowski (PQShield, LTD), 1/31/2025

I have the following comments for Special Publication 800-56Cr2:

1. Section 2 permits using a "hybrid" share the secret of the form Z'=Z||T, where shared secret Z is generated by approved key-establishment followed by T - an auxiliary shared secret. According to the current wording in the document, the order of those shared secrets matters.

Enforcing a strict order complicates the integration of hybrid schemes into protocols. Ideally, such integration should remain unchanged even if one of the key establishment methods is later disapproved. Allowing arbitrary ordering provides greater flexibility, ensuring that protocol integration remains resilient to changes resulting from the deprecation of key exchange algorithms.

2. Section 5.3 describes the extract-then-expand key derivation function (KDF), where the expansion step is performed multiple times. It includes the following requirement for the multi-expansion step:

"The derived keying material, DerivedKeyingMaterial1, DerivedKeyingMaterial2, … , and DerivedKeyingMaterialm shall not be output until all m of the bit strings have been successfully computed. If an error occurs during randomness extraction or key expansion, then this key-derivation method shall not output any derived keying material."

This requirement appears to impose an unusual constraint on KDF implementations, potentially making them impractical to use in certain scenarios. One implementation of extract-then-multi-expand is HKDF, which typically follows an interface composed of three functions:

* HKDF_init(ctx, ...) – Initializes the context (ctx), which maintains the state of the operation across function calls.

* HKDF_extract(ctx, ...) – Performs the extraction step.

* HKDF_expand(ctx, ...) – Executes the expansion step, which may be called multiple times.

To comply with NIST's requirements, this interface would need to be modified as follows: * The number of future calls must be stored in the ctx.

* The ctx must also store all DerivedKeyingMaterial_n values computed during HKDF_expand(). This introduces additional storage requirements, making the HKDF implementation less generic.

* The HKDF_expand() API must be modified so that it does not immediately output the computed value.

* An additional API function would be required to output all m derived keying material values at once.

Such modifications, particularly for hardware-based implementations, could complicate deployment within existing software stacks. For example, this change would impact implementations of TLS 1.3 (as specified in RFC 8446, Section 7.1).

Furthermore, it is possible to envision protocols where DerivedKeyingMaterial_1 is required before computing DerivedKeyingMaterial_2. For instance, two parties might exchange data between the computation of the first and second keys, using the exchanged data as input for the second key derivation. Under the current SP 800-56C Rev. 2 constraints, a compliant multi-expansion HKDF cannot be used in such applications.

3. As KMAC has now been approved, the note regarding it in Section 8.3 may be removed.

4. As a minor note, Sections 5.1 and 5.2 mention that HMAC will hash the salt if it exceeds the input block size of the hash function. This information appears to be unnecessarily repeated (potentially).

9. Comments Máté Horváth (University of Wuppertal), 1/31/2025

I would like to submit the following comment on the document NIST SP 800-56Cr2, Recommendation for Key-Derivation Methods in Key-establishment Schemes.

[1] ``permits the use of a `hybrid' shared secret of the form Z'= Z || T, a concatenation consisting of a `standard' shared secret Z that was generated during the execution of a key-establishment scheme (as currently specified in [SP 800-56A] or [SP 800-56B]) followed by an auxiliary shared secret T that has been generated using some other method. /.../ The key-derivation methods specified in this Recommendation will process a hybrid Z′ in the same way they process a standard Z.'' (See Section 2 in [1]).

This recommendation is especially important as post-quantum transition will likely involve hybrid KEMs. After the recent standardization of ML-KEM [6], I think expecting CCA security from hybrid solutions is natural.

Unfortunately, the security of a hybrid KEM (combining KEM1 and KEM2 that output keys Z and T respectively) that derives the shared secret in the recommended form (i.e., concatenating the shared secrets, obtained from KEM1 and KEM2) crucially depends on some implementation choices of the underlying key derivation function (KDF) that are not clearly guided by [1]. Namely, if the FixedInfo input of the used KDF includes ciphertexts c1 and c2 of KEM1 and KEM2, then the resulting hybrid KEM is IND-CCA secure (see [2,3]), otherwise, it is only IND-CPA secure (see [3] and the CCA attack described in [1,3,4]). Currently, the recommendation about choosing FixedInput lists several context-specific information that may be appropriate for inclusion. The list in [5] contains ``public and/or private information shared between parties U and V before or during the transaction, such as nonces or pre-shared secrets''. While c1 and c2 would fit into this recommendation, most probably the importance of this choice (namely that it affects whether CCA or only CPA security is achieved) is unclear for practitioners who are relying on these documents.

The following two possible solution to the problem have the same effect:

(a) explicitly requiring the inclusion of information shared between parties (in particular ciphertexts in case of hybrid KEMs) to the FixedInfo input of the KDF in [1], without changing the recommendation about forming the hybrid shared secret;

(b) changing the recommendation about forming the hybrid shared secret in the form Z'= Z || T||c1||c2, without modifying the instructions about the FixedInfo input of the used KDF.

Solution (a) is insufficient because it would also affect non-hybrid solutions, where such change is unnecessary. Solution (b) also does not seem to be entirely satisfactory because there is a trade-off here. While including more information to the KDF input increases security, it also affects the performance negatively. In such cases general solutions are usually not the best. Indeed, [4] showed that combining ML-KEM-768 and the ECDH over the X25519 curve by creating the shared secret in the form Z'= label||k1||k2||pk2||c2 (where label is a fixed 6 byte string, k1 is the shared secret obtained from ML-KEM, and k2, pk2, c2 are the shared key, public key, and ciphertext belonging to ECDH) also implies IND-CCA security of the hybrid KEM. At the same time, this solution is also more efficient than the general solution, because the ML-KEM ciphertext (which is longer than pk2) does not have to be included into the KDF input. Moreover, if my understanding is correct, the result of [4] also extends to NIST curves P-256, P-384, P-521 [7], and to Curve25519 and Curve448 [8] as all of these are Nominal Groups [9] and to all security levels of ML-KEM (as the proof of ML-KEM's collision resistance seems to apply to all security levels). On the other hand, the solution of [4] does not extend to general KEM hybrids.

To conclude, I find the current recommendation about hybrid shared secrets unsatisfactorily because CCA security of the resulting hybrid KEM is not ensured even if one follows all the guidelines. At the moment, the solution of [4] seems to be the best for me because it seems to capture the hybrid of solutions that are already approved by NIST [5,6] (but I am not familiar with [5], and also only spent limited time understanding [4]). For other type of hybrids, I am not aware of better solution then (b).

References

[1] Elaine Barker, Lily Chen, and Richard Davis. Recommendation for key-derivation methods in key-establishment schemes. NIST Special Publication (NIST SP 800-56C Rev. 2), 2020.https://doi.org/10.6028/NIST.SP.800-56Cr2.

[2] Federico Giacon, Felix Heuer, and Bertram Poettering. KEM combiners. In Public Key Cryptography (1), volume 10769 of Lecture Notes in Computer Science, pages 190–218. Springer, 2018.

[3] Adam Petcher and Matthew Campagna. Security of hybrid key establishment using concatenation. Cryptology ePrint Archive, Paper 2023/972, 2023.

[4] Manuel Barbosa, Deirdre Connolly, Joao Diogo Duarte, Aaron Kaiser, Peter Schwabe, Karoline Varner, and Bas Westerbaan. X-wing: The hybrid KEM you've been looking for. Cryptology ePrint Archive, Paper 2024/039, 2024.

[5] Elaine Barker, Lily Chen, Allen Roginsky, Apostol Vassilev, and Richard Davis. Recommendation for pair-wise key-establishment schemes using discrete logarithm cryptography. NIST Special Publication (NIST SP 800-56A Rev. 3), 2018. https://doi.org/10.6028/NIST.SP.800-56Ar3

[6] National Institute of Standards and Technology. Module-lattice-based key-encapsulation mechanism standard. Technical Report Federal Information Processing Standards Publication (FIPS PUBS) 203, U.S. Department of Commerce, Washington, D.C., 2024. DOI:
https://doi.org/10.6028/NIST.FIPS.203

[7] National Institute of Standards andTechnology: Digital Signature Standard (DSS). FIPS Publication 186-4 (Jul 2013), https://doi.org/10.6028/nist.fips.186-4

[8] Langley, A., Hamburg, M., Turner, S.: Elliptic curves for security. RFC 7748, RFC Editor (Jan2016), https://www.rfc-editor.org/rfc/rfc7748.html

[9] Joël Alwen, Bruno Blanchet, Eduard Hauck, Eike Kiltz, Benjamin Lipp, and Doreen Riepel. Analysing the HPKE standard. In Anne Canteaut and François-Xavier Standaert, editors, Advances in Cryptology – EUROCRYPT 2021, Part I, volume 12696 of LNCS, pages 87–116, 2021. Springer, Heidelberg, Germany. doi: 10.1007/978-3-030-77870-5_4.