Public Comments on SP 800-38C

Comment period: June 14, 2024 to September 13, 2024

On June 14, 2024, NIST's Crypto Publication Review Board initiated a review of SP 800-38C: *Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality*.

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 Christopher Dickerman, June 21, 2024

Hello,

On page 10 of SP 800-38C Section 6.2, Step 9, there appears to be a typo on the index of Y:

Y_j = CIPH_K(B_i \xor Y_{i-1})

Should be

Y_i = CIPH_K(B_i \xor Y_{i-1})

where the subscript "j" should be changed to "i".

Best,

Christopher Dickerman

2. Comments from Joachim Vandersmissen (atsec information security corporation), July 5, 2024

Dear NIST,

Thank you for soliciting feedback from the community regarding the proposed changes to SP 800-38C. I personally don't have any preference regarding a 64-bit authentication tag size. Regardless of NIST's decision, it might be more appropriate if the authentication tag size guidelines for all applicable algorithms (CMAC, CCM, HMAC, KMAC, GMAC, GCM, ...) are specified in a central place, similar to SP 800-131Ar2. This avoids any inconsistencies between the individual algorithm standards and provides an easy reference.

In Section 5.3, the standard states "The nonce is not required to be random". Could this be clarified by stating "The nonce is not required to be unpredictable", to be consistent with the terminology used in SP 800-38A?

In Section 6.1, the output (ciphertext) is defined as the concatenation of the encrypted plaintext and the MAC tag. For consistency with SP 800-38D, and to match real-world implementations more closely, it would be convenient to have the output defined as a tuple consisting of the encrypted plaintext and the MAC tag. This would require rewriting the algorithm in Section 6.2 too, as the input would now be a tuple of a ciphertext and a MAC tag, and the MSB/LSB functions are no longer used. Nevertheless, I think these changes could be worthwhile from an implementation standpoint.

In Section 6.2, step 9 should state Y_i instead of Y_j.

The last paragraph of Section 6.2 contains a "shall" statement related to preventing side-channel attacks. This is a very harsh requirement. In my experience, NIST algorithm specifications generally don't include any "shall" statements related to side-channel analysis, even where side channel attacks can be much more devastating (e.g., ECDSA nonce generation). This is not to say that the requirement is invalid, however it does seem out of place here. For consistency with other standards (SP 800-38A, B, D, ...), I suggest that the "shall" statement is changed to a "should" statement.

If my suggestion to change the output of the algorithm in Section 6.1 is applied, Appendix B.1 should be updated to reflect the change (e.g., "For any purported ciphertext that is at least Tlen bits long, the rightmost Tlen bits correspond to an encrypted MAC, and the remaining bits correspond to an encrypted payload.")

In Appendix E, a few reference links to are broken or redirects:
[1] redirects to the Modes Development page, but this page does not contain a link to the reference.
[7] should be updated to reflect the fact that SP 800-38B has been published.

Kind regards,
Joachim Vandersmissen

--
Joachim Vandersmissen
joachim@atsec.com
atsec information security corporation

3. Comments from Michael Bowler (Synopsys), September 6, 2024

Dear Crypto Publication Review Board,

Please see attached document for comments from Synopsys on the SP 800-38C publication.

Best regards,

Michael Bowler
Security HW and System Architect
Synopsys Inc.

Dear Crypto Publication Review Board,

Synopsys would like to thank NIST for the opportunity to register our comments with respect to SP 800-38C (The CCM Mode for Authentication and Confidentiality).

We find the lack of explicit specification of the formatting function and counter generation functions within SP 800-38C problematic. SP 800-38C gives examples of compliant formatting and counter generation functions, however the actual implementation is left up to the user. Generic implementations of CCM (for example, a hardware IP core) either have to make assumptions of the target use case or offload these functions to the integrator. Integrators may not have sufficient security expertise, resulting in misunderstanding and misuse. Misunderstanding of the CCM specification has resulted in numerous support cases from users of our AES-CCM hardware IP cores. Misuse can result in real world security issues.

Contrast SP 800-38C to SP 800-38D (GCM). GCM defines the input formatting and counter generation functions explicitly. The user simply provides inputs of Key, IV, additional authentication data, and plaintext/ciphertext data. The algorithm produces the result. The user is not required to understand the specification, other than some simple rules with respect to input length constraints. With CCM the user is given a complex task to pre-format the input data with "an algorithm of their choice" that meets the security requirements outlined in SP 800-38C chapter 5.4, and concurrently come up with a counter generation function that does not cause a counter collision with B_0 (rule #3 of chapter 5.4).

Our suggestions:

1) Define the input formatting function explicitly.
    a. The example in Appendix A of SP 800-38C (or alternatively RFC-3610) is a good start.
    b. We suggest the values of q and n are restricted to a small subset of approved values. e.g. {q, n} inside the set of {{2,13}, {4, 11}} would be a good compromise to support existing standards (e.g. IEEE 802.11, which uses {q, n} == {2, 13}, and future use with larger packets where {q, n} == {4, 11})
    c. We believe the constraint of max(q) == 4, corresponding to 2^32 bytes of payload, should not be overly restrictive for implementations of packet-based protocols. Quoting from chapter 3 of SP 800-38C: *"CCM is intended for use in a packet environment, i.e., where all data is in storage before CCM is applied"*

d. With n >= 11, use of a key would be limited to at least 2^{88} packets, which we believe is sufficient for to all real-world applications of packet-based protocols.

2) Define the counter generation function explicitly.
    a. The example in Appendix A of SP 800-38C is a good start.
    b. Applying the constraint of q <=4, allows CCM to share the same counter increment function as GCM, which is useful for hardware implementations which implement both standards.

Best regards,

Michael Bowler
Security HW and System Architect
Synopsys Inc.

4. Comments from John Preuß Mattsson (Ericsson), September 14, 2024

Dear NIST,

Thanks for your continuous efforts to produce well-written open-access security documents. Please find attached our comments on [SP 800-38B and] 800-38C.

Best Regards,
John Preuß Mattsson,
Expert Cryptographic Algorithms and Security Protocols

ERICSSON

Date: September 14, 2024

Ericsson AB
Group Function Technology
SE-164 80 Stockholm
SWEDEN

Comments on SP 800-38B "The CMAC Mode for Authentication" and SP 800-38C "The CCM Mode for Authentication and Confidentiality"

Dear NIST,

Thanks for your continuous efforts to produce well-written, user-friendly, and open-access security documents.

Please find below our feedback on the guidance for authentication tag lengths:

As explained in [1], short tags have many important applications in various industries. For instance, 32-bit tags are standard in most radio link layers including 5G, 64-bit tags are widely used in transport and application layers of the Internet of Things (IoT), and 32- and 64-bit tags are common in media protection applications. Although GMAC and GCM are not suitable for short tags [2], CMAC and CCM are perfectly suitable in these scenarios. For many applications, short tags provide substantial benefits and are secure in practice:

- Audio packets are small, numerous, and ephemeral, so on the one hand, they are highly sensitive to cryptographic overhead, and on the other hand, forgery of individual packets is not a big concern as it typically is barely noticeable. Real-time audio codecs commonly encodes only 20 ms of audio in each packet.

- Constrained radio networks are not only characterized by very small frame sizes in the range of tens of bytes transmitted a few times per day at ultra-low speeds, but also high latency, and severe duty cycle constraints. In multi-hop networks, the already small frame sizes are further reduced for each additional hop [3]. To achieve a 50% chance of a single forgery using a 64-bit ideal MAC, an attacker would need to transmit one billion packets per second for 300 years. This is completely unfeasible for constrained radio systems and the chance of this happening is negligible compared to the risk of data corruption due to hardware failure or cosmic rays [4].

- In lockstep protocols like EDHOC [5], where the session is terminated if verification fails, security against online attacks is given by the sum of the strength of the verified MACs. I.e., n 64-bit tags

provide the same security as a single $64n$-bit tag. Similar more complex uses of short tags are common in IoT and known as aggregate, compound, or cumulative MACs, see e.g., [6].

– In 5G user plane [7], encryption provides confidentiality for 5G Voice and privacy-sensitive metadata such as IP addresses and domain names. Integrity protection provides IND-CCA confidentiality and protects IP addresses from modification. 32-bit tags offer sufficient protection in this case. Applications with sensitive data use end-to-end over-the-top protection with strong security protocols like TLS.

After careful analysis of the risks, we think short tags should be allowed for:
– Real-time audio packets, where a few single forgeries do not alter the meaning.
– Layer 2 in networks where sensitive data is already protected on higher layers.
– Protocols where the security is given by the sum of the tag lengths.
– Constrained radio networks, where the low bandwidth preclude many repeated trials.

We therefore do not think the CMAC and CCM publications should require that all authentication tags meet a minimum threshold, such as 64 bits, but requiring at least 32-bit tags would be acceptable. For all applications of short tags it is essential that the MAC behaves like an ideal MAC, i.e., the forgery probability is $\approx 2^{-Tlen}$, even after many generated MACs, many forgery attempts, and after a successful forgery, where $Tlen$ is the bit length of the authentication tag. This implies that the expected number of forgeries is $\approx v \cdot 2^{-Tlen}$, where v is the number of verification queries. CMAC and CCM behave like ideal MACs when the tag length is much shorter than the block size and the number of queries are limited.

Tag length alone is not an accurate measure of security, as standardized MAC algorithms deviate from an ideal MAC in various ways. Rather than recommending a tag length of 64-bits, NIST should recommend constrains that ensure an effective tag length of 64 bits, i.e., security equal to or stronger than that of a 64-bit ideal MAC. This provides high security in almost all applications and implies that the expected number of forgeries is $\leq v \cdot 2^{-64}$. BSI states that using an ideal MAC with 96-bit tag length is acceptable [8] but achieving this level of security with AES-CMAC, AES-CCM, AES-GMAC, or AES-GCM is almost impossible since the security is limited by the narrow 128-bit block size in AES and the 128-bit digest size in GHASH. For untruncated tags, the forgery probability in all four algorithms depends on both the message length and the block length; in CMAC and CCM, the integrity advantage is quadratic in the number of queries, and all four algorithms offer zero reforgeability resistance.

Please find below our comments on SP 800-38B (CMAC):

- The specification states that CMAC provides assurance of authenticity. The revision should explain in more detail what assurances CMAC provides, for example that it does not provide key commitment. Most readers of SP 800-38B cannot be expected to know exactly what properties a message authentication code (MAC) algorithm typically provides. Users and implementors of cryptography are known to assume non-existing properties like key commitment, leading to practical vulnerabilities. We strongly agree with NIST's guidelines that cryptographic standards should be chosen to minimize the demands on users and implementers as well as the adverse consequences of human mistakes and equipment failures [9].

- The revision should specify that CMAC provides Strong Existential Unforgeability under Chosen-Message Attack (SUF-CMA) and shortly explain what this means.

- The specification defines the term least significant bit and the symbol $LSB_s(X)$, but they are never used. The revision should remove them.

- The revision should explain that CMAC offers very different security properties depending on whether the tags are truncated or not. For truncated tags with a small number of generation queries q, CMAC behaves like an ideal MAC and therefore provides reforgeability resistance. In this case, the integrity advantage and the expected number of forgeries are both linear in v. For non-truncated tags, where $Tlen$ equals the block size b, the integrity advantage is quadratic in q, the expected number of forgeries is cubic in the number of queries, and CMAC does not offer any reforgeability resistance.

- Following the withdrawal of TDEA [10], the revision should remove TDEA and 64-bit blocks. Given that NIST will hopefully approve Rijndael-256-256 in the near future [11], the revision should define the bit string R_{256} and discuss constraints for 256-bit block sizes. The security of AES-CMAC is severely limited by the narrow 128-bit block size in AES. The limitation does not only restrict the output length [12], but also affects collision attack complexity, integrity advantage, expected number of forgeries, and reforgeability resistance.

- Section 5.3 states that: *"Any intermediate value in the computation of the subkey, in particular, $CIPH_K(0^b)$, shall also be secret."*. This gives the impression that using e.g., $\mathrm{E_K}(0101\ldots)$ as a Key Check Value (KCV) might be ok, even if there is no reason to believe that such use is secure.

  The CMAC generation function described in Section 2.1 can be written as the equation

$$T = E_K(E_K(\ldots E_K(E_K(M_1) \oplus M_2) \ldots \oplus M_{n-1}) \oplus M_n$$

  Assume an attacker knows $X_1, X_2, \ldots X_m$ and $E_K(X_1), E_K(X_2) \ldots E_K(X_m)$ and let

$$Y_{ij} = E_K(X_i) \oplus X_j$$
$$Z_i = E_K(X_i) \oplus B$$

  where B is any chosen block. Then the messages

$$B \,||\, S,\ \ X_i \,||\, Z_i \,||\, S,\ \ \ X_i \,||\, Y_{ij} \,||\, Z_j \,||\, S,\ \ X_i \,||\, Y_{ij} \,||\, Y_{jk} \,||\, Z_k \,||\, S,\ \ldots$$

  where $i, j, k, \ldots \in \{0,1, \ldots, m\}$ and S is a suffix of any length (not necessarily complete blocks) have identical tags. Making any X, $E_K(X)$ pair public completely breaks the security of CMAC. Furthermore, making s bits of any X, $E_K(X)$ pair public lowers the security of untruncated tags with s bits. There is nothing particular with $E_K(0^b)$. The attack above is a significant improvement of [13]. We plan to soon publish this result on IACR ePrint.

  The revision should state that making (part of) any X, $E_K(X)$ pair public breaks the security of CMAC.

- Decrypting each side of the CMAC equation $m - 1$ times while moving the M_j operands to the left-hand side, rearranging, and letting $i = n - m$, we can write any message block as a function of the other message blocks and the tag

$$M_i = E_K(E_K(\dots E_K(E_K(M_1) \oplus M_2) \dots \oplus M_{i-2}) \oplus M_{i-1}) \oplus D_K(D_K(\dots D_K(D_K(T) \oplus M_n) \dots \oplus M_{i+2}) \oplus M_{i+1})$$

  With access to adaptive encryption and decryption oracles, an attacker can freely choose a prefix $P = M_1 \,||\, \dots \,||\, M_{i-1}$ and a suffix $S = M_{i+1} \,||\, \dots \,||\, M_n^*$ and then calculate M_i so that the message produces a chosen tag T. The attacker can control all bits except for the random-looking M_i. This is a significant improvement of the attack by Amazon described in SP 800-108 [14] and shows that CMAC does not offer any preimage or collision-resistance when the key is known. We plan to soon publish this result on IACR ePrint. As acknowledged by NIST [14], lack of preimage or collision-resistance creates problems when used as a KDF [12,14] or as an entropy conditioning function [15], since a party controlling the inputs might be able make keys predictable or (partially) identical [14,16,17]. We think collision- and preimage-resistance should be a requirement for a Key Derivation Function (KDF). The potential solution to the Amazon attack described in SP 800-108 [14] is promising but seems ad-hoc and does not come with any motivation. We do not think CMAC should be used as a KDF or as an entropy conditioning function unless it can be proven that a specific construction and formatting of the inputs ensures collision resistance and preimage resistance.

  The revision should state that with a known key, CMAC does not offer any preimage or collision-resistance and is therefore unsuitable as a KDF.

- The revision should address Rogaway's comments in Section 8.6 of [18].

- *"The length of T, denoted Tlen, is a parameter that shall be fixed for all invocations of CMAC with the given key."*

  This kind of formulation should also be added to 800-38C and 800-38D, but a much better formulation would be that a minimum length of T should be fixed for all invocations. There is no problem with variable length tags as long as a minimum length is enforced. See e.g., Robust AE (RAE) [19] for an excellent use of variable length tags.

- The specification discusses tag length and message span in different appendixes as if they were independent. Let $Tlen$ be the tag length in bits, b be the block size in bits, ℓ the message length in blocks, q the number of generation queries, and v the number of verification queries. Then based on [20] and ignoring constant factors, our understanding is that the integrity advantage is bounded by $\approx v \,/\, 2^{Tlen} + q^2 \,/\, 2^b + q\ell^2 \,/\, 2^b$, where the terms $v \,/\, 2^{Tlen}$ and $q^2 \,/\, 2^b$ correspond to known attacks and are tight. The security is therefore a function of both the tag length and the message span q and the revision should discuss them together.

- Appendix A.1 discusses guessing attacks and states:

  "In particular, if the attacker selects a MAC at random from the set of strings of length Tlen bits, then the probability is 1 in 2^{Tlen} that the MAC will be valid. Consequently, larger values of

Tlen provide greater protection against such an event. Of course, an attacker may attempt to systematically guess many different MACs for a message, or for different messages, and thereby increase the probability that one (or more) of them will be accepted as valid. For this reason, a system should limit the number of unsuccessful verification attempts for each key.

This recommendation is lacking for several reasons. For many tag lengths and message spans, a guessing attack is not the best-known attack, and larger tags do not provide better security. The stated probability 2^{-Tlen} might make readers believe that CMAC behaves like an ideal MAC. NIST does not state what the system should do when the limit is reached. Not accepting more verification attempts makes the system vulnerable to denial-of-service attacks and lowers the availability of the system. Rekeying does not help at all against the attack NIST describes as the forgery probability for each attempt would remain 2^{-Tlen}. We suggest that the revision removes the quoted text and all other text on guessing attacks. Any text on limits should describe exactly what to do when the limit is reached and how this increases security.

As long as $\ell^2 \leq q$ our understanding is that the integrity advantage is $\approx v \,/\, 2^{Tlen} + q^2 \,/\, 2^b$ and as a full tag collision lets an attacker create an unlimited amount of forgeries [21], the expected number of forgeries is $\approx v \,/\, 2^{Tlen} + vq^2 \,/\, 2^b$. An ideal MAC would have integrity advantage $v \,/\, 2^{Tlen}$ and expected number of forgeries $v \,/\, 2^{Tlen}$. AES-CMAC with 32-bit tags and small q as used in 4G and 5G always behaves like an ideal MAC. AES-CMAC with 128-bit tags never behaves like an ideal MAC. We note that Rijndael-256-256 with 128-bit tags would behave like an ideal MAC. The revision should describe that CMAC only behaves like an ideal MAC when $q^2 \cdot 2^{Tlen} < 2^b$.

For narrow block sizes, larger values of $Tlen$ only provides greater assurance for small values of q. For AES-CMAC with $q = 2^{48}$, which NIST recommends, the integrity advantage for $Tlen = 32$ and $Tlen = 128$ are approximately the same. The revision should explain this.

- Appendix A.2 states: *"A value of Tlen that is less than 64 shall only be used in conjunction with a careful analysis of the risks of accepting an inauthentic message as authentic.".* The revision should mention that there are protocols where the security is given by the sum of the tag lengths.

- Appendix A.2 states:

  "a value of Tlen smaller than 64 should not be used unless the controlling protocol or system sufficiently restricts the number of times that the verification process can return INVALID, across all implementations with any given key. For example, the short duration of a session."

  "Tlen should satisfy the following inequality: Tlen ≥ lg(MaxInvalid / Risk)"

  Ensuring a high security per key is a theoretical simplification that does not reflect practical security. Rekeying based on MaxInvalid does not increase practical security as an attacker will just continue to attack subsequent keys. Not accepting more verification attempts without rekeying enables denial-of-service attacks on the system. Similarly, ensuring a high security for a single session does not help if there are many sessions. The formula does not consider that Risk depends on the message span q and the block size b. In the revision we think Appendix A and Appendix B should be merged and much of the content rewritten.

- Appendix B states that:

  *"an attacker **may** be able to exploit a collision to produce the valid MAC for **a** new message"*

  This seems to severely underestimate the problems with collision attacks and multiple forgeries. Given a collision, an attacker can trivially forge any number of new messages with probability 1. See the analysis of McGrew and Fluhrer [21]. Suggestion:

  "an attacker can exploit a collision to produce the valid MAC for any number of new messages"

- Appendix B states that:

  "For general-purpose applications, the default recommendation is to limit the key to no more than 2^{48} messages when the block size of the underlying block cipher is 128 bits, as with the AES algorithm, and 2^{21} messages when the block size is 64 bits, as with TDEA. Within these limits, the probability that a collision will occur is expected to be less than one in a billion for the AES algorithm, and less than one in a million for TDEA"

  This recommendation is lacking for several reasons. Collisions are not interesting for readers and there is no reason to switch from binary to decimal. The interesting property for users and developers is forgeries and the forgery probability can be much higher that the collision probability. If the message span is q, the collision probability is $\approx q^2 \,/\, 2^b$, the forgery probability is $\approx MaxInvalid \,/\, 2^t + q^2 \,/\, 2^b$, and the expected number of forgeries are $\approx vq^2 \,/\, 2^b$. The specification does not recommend any restriction on the number of verification queries, but even if both q and v is limited to 2^{48}, the expected number of forgeries on AES-CMAC is $\approx 2^{16}$. We suggest that q and v are severely limited. If we compare CMAC with an ideal MAC, the effective tag length is $\approx \log_2(2^b / q^2)$. We do not know if the term $\approx q\ell^2 \,/\, 2^b$ corresponds to concrete attacks and if it can be used for reforgeability. We think NIST in all algorithm specifications should recommend parameters giving an effective tag length of at least 64 bits. As a comparison, the effective tag length of AES-GMAC is $\approx \log_2(2^{128} / v\ell)$.

- Appendix C states that a protocol or application may protect against replay attacks. This is weaker that the text in SP 800-38C that says that a protocol or application should protect against replay attacks. We think replay protection should be a strong requirement unless careful analysis of the whole system shows that replay protection is not needed in some specific part. Users and developers expect replay protection and higher layer protocols are often designed with the expectation that the security protocol provides replay protection. One example where no replay protection might be acceptable is 0-RTT in TLS 1.3 [22], but only after very careful analysis by the server on what kind of data to accept. Systems without replay protection often leads to surprising attacks and are hard to analyze. It is not a coincidence that Section 8 and E.5 in RFC 8446 [22] spends many pages analyzing security considerations for replayable 0-RTT data.

  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. It is sometimes argued that replay protection can be turned off in a lower layer as it is handled in a higher layer. This might be correct for the current configuration but often leads to security vulnerabilities when the higher layers are updated or changes. One recent example is DTLS 1.3 [23] where the post-handshake messages were designed with the expectation that replay protection is provided by the record layer and it was forgotten that DTLS allows turning of record layer replay protection. IETF is discussing updating DTLS 1.3 to forbid turning off replay protection.

The revision should mandate replay protection unless careful analysis of the whole system shows that replay protection is not needed in some specific part.

Please find below our comments on SP 800-38C (CCM):

- Many of our comments on SP 800-38B also apply to 800-38C. Below are additional comments.

- Rogaway states in [18] that:

  "I question NIST's decision to "isolate" the formatting and counter-generating functions to a non-binding appendix; as ungainly as these functions may be, I believe that their selection must, in the interest of interoperability and assurance, be considered as an intrinsic part of the standard, something that is fixed and mandated in any follow-on standardization"

  "to the best of my knowledge, no formatting or counter-generation functions other than the canonical pair have ever been specified or used. In addition, all of the other standards specifying CCM, including RFC 3610 [204], fold the definition of the canonical formatting function and counter-generation function directly into the scheme that they too name CCM. In general, given the relatively complex requirements that the (Format, Count) pair must satisfy, given the complex engineering-tradeoffs implicit in selecting a definition for these objects, given the easy opportunity for error, and given that the goal of interoperability that is one of the aims of a cryptographic standard, I find it untenable to think that CCM is ever going to be used with anything other than the canonical formatting function and counter-generation function."

  We agree with Rogaway, the revision should mandate the formatting and counter-generating functions and move them to the body of the specification.

- Section 5.3 states *"The nonce is not required to be random",* suggesting that random nonces is allowed. If CCM is used with random nonces, the security against ciphertext-only collision attacks compromising confidentiality is only $\approx |IV| + 1 - \log_2 n$ where $|IV|$ can be as low as 56 and n can be as large as $\approx 2^{59}$, see Table 1 and Section 3.1 of [24]. Unlike 800-38D, 800-38C does not mandate any specific nonce constructions, maximum collision probabilities, or maximum number of invocations.

  As stated in Section 11.9 of [18], Rogaway and Fergusson suggest that "The nonce is not required to be random" should be interpreted as the nonce need not be unpredictable. It is likely this was NIST's intention. However, we believe that developers and users are unlikely to interpret the

statement in this way. In fact, the official documentation of many cryptographic libraries exemplifies the use of AES-CCM with random nonces and the widely-used Python package PyCryptodome [25] defaults to 11-byte random nonces with AES-CCM, resulting in only $89 - log_2 n$ bits of security.

The revision should explicitly forbid random nonces. If random nonces are allowed they should at least have the same restriction as in GCM, where the random field is at least 96 bits and the number of GCM invocations is restricted to 2^{32}. Note that we do not think a 96-bit random field provides acceptable security and should be forbidden. Rogaway makes the same recommendation *"I also recommend that randomly-chosen IVs be explicitly disallowed"* [18].

- Section 5.1 states that "The total number of invocations of the block cipher algorithm during the lifetime of the key shall be limited to 2^{61}"

  As written this this limit apply to the total number of invocations in the encryption and decryption function. This is not very useful for users and developers. The revision should state independent limits for the encryption and decryption function. As described by Rogaway in Section 11.8 of [18] the relation between the number of block cipher invocations and the number of CCM invocations is complicated. For simplicity, implementations might want to limit the number of CCM invocations instead of block cipher invocations. The revision should give guidance to users and developers that are limiting the number of CCM invocations.

Best Regards,
John Preuß Mattsson,
Expert Cryptographic Algorithms and Security Protocols

[1] Campagna, Maximov, Preuß Mattsson, "Galois Counter Mode with Secure Short Tags (GCM-SST)"
https://csrc.nist.gov/csrc/media/Events/2023/third-workshop-on-block-cipher-modes-of-operation/documents/accepted-papers/Galois%20Counter%20Mode%20with%20Secure%20Short%20Tags.pdf

[2] Mattsson, Westerlund, "Authentication Key Recovery on Galois/Counter Mode (GCM)"
https://eprint.iacr.org/2015/477.pdf

[3] Preuß Mattsson, Selander, Smeets, Thormarker, "Constrained Radio Networks, Small Ciphertexts, Signatures, and Non-Interactive Key Exchange"
https://eprint.iacr.org/2023/913.pdf

[4] Preuß Mattsson, "Hidden Stream Ciphers and TMTO Attacks on TLS 1.3, DTLS 1.3, QUIC, and Signal"
https://eprint.iacr.org/2023/913.pdf

[5] IETF, "Ephemeral Diffie-Hellman Over COSE (EDHOC)"
https://eprint.iacr.org/2023/913.pdf

[6] Wagner, Serror, Wehrle, Henze, "When and How to Aggregate Message Authentication Codes on Lossy Channels?"
https://eprint.iacr.org/2023/1919.pdf

[7] 3GPP TS 33.501, "Security architecture and procedures for 5G System"
https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3169

[8] BSI, "Cryptographic Mechanisms: Recommendations and Key Lengths"
https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf?__blob=publicationFile

[9] NIST, "NIST Cryptographic Standards and Guidelines Development Process"
https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=920594

[10] NIST, "NIST to Withdraw Special Publication 800-67 Revision 2"
https://csrc.nist.gov/news/2023/nist-to-withdraw-sp-800-67-rev-2

[11] NIST, "NIST Options in for Encryption Algorithms and Modes of Operation"
https://csrc.nist.gov/csrc/media/Presentations/2024/options-for-encryption-algorithms-and-modes/images-media/sess-3-regenscheid-acm-workshop-2024.pdf

[12] NIST, "Recommendation for Key-Derivation Methods in Key-Establishment Schemes"
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Cr2.pdf

[13] Iwata, Wang, "Impact of ANSI X9.24-1:2009 Key Check Value on ISO/IEC 9797-1:2011 MACs"
https://www.iacr.org/archive/fse2014/85400153/85400153.pdf

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

[15] NIST SP 800-90C, "Recommendation for Random Bit Generator (RBG) Constructions"
https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90C.4pd.pdf

[16] Fischlin, Janson, Mazaheri, "Backdoored Hash Functions: Immunizing HMAC and HKDF"
https://eprint.iacr.org/2018/362.pdf

[17] Backendal, Bellare, Günther, Scarlata, "When Messages are Keys: Is HMAC a dual-PRF?"
https://eprint.iacr.org/2023/861.pdf

[18] Rogaway, " Evaluation of Some Blockcipher Modes of Operation"
https://www.cs.ucdavis.edu/~rogaway/papers/modes.pdf

[19] Hoang, Krovetz, Rogaway, "Robust Authenticated-Encryption: AEZ and the Problem that it Solves"
https://eprint.iacr.org/2014/793.pdf

[20] Chattopadhyay, Jha, Nandi, "Towards Tight Security Bounds for OMAC, XCBC and TMAC"
https://eprint.iacr.org/2022/1234.pdf

[21] McGrew, Fluhrer, "Multiple forgery attacks against Message Authentication Codes"
https://eprint.iacr.org/2005/161.pdf

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

[23] IETF RFC 9147, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3"
https://www.rfc-editor.org/rfc/rfc9147.html

[24] Preuß Mattsson, "Collision Attacks on Galois/Counter Mode (GCM)"
https://eprint.iacr.org/2024/1111.pdf

[25] PyCryptodome Documentation
https://pycryptodome.readthedocs.io/en/latest/src/cipher/modern.html#ccm-mode