ERICSSON

Date: March 14, 2025

Ericsson AB
Group Function Technology
SE-164 80 Stockholm
SWEDEN

Comments on NIST SP 800-38D
GCM and GMAC Block Cipher Modes of Operation

Dear NIST,

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

Please find below Ericsson's comments on NIST SP 800-38D [1]. In addition to our feedback on the approaches outlined in the pre-draft call for comments [2], we provide several new observations identified since our previous comments on SP 800-38D [3–4]. One of these is that GCM tags do not behave like ideal tags, as the forgery probability depends on the lengths of the plaintext and associated data as well as prior forgery attempts. Furthermore, stricter constraints are necessary to mitigate risks associated with nonce collisions attacks as well as attacks using the absence of block collisions.

We recommend that NIST standardizes Galois Counter Mode with Strong Secure Tags (GCM-SST) [5] including Rijndael-GCM-SST, a general function for deriving key-nonce pairs that can be instantiated with different KDFs and AEADs, and that stricter constraints are added to SP 800-38D.

In the comments we will use the following terms defined in [5–6]:

- P_MAX, the maximum length of plaintext in bytes
- A_MAX, the maximum length of associated data in bytes
- Q_MAX, the maximum number of invocations of the encryption function
- V_MAX, the maximum number of invocations of the decryption function

Function for deriving key-nonce pairs:

- We have briefly looked at AES-XGCM [7], XAES-GCM [8], and DNDK-GCM [9–10]. The differences are supported key lengths, the KDF used, the length of the random nonce (192-bit or 224-bit), whether the whole nonce is used in the key derivation, and optional key commitment.

  AES-XGCM: Uses a 16-byte random value to derive the AEAD key. The 12-byte random nonce N is not used in the key derivation. We like the idea of a separate nonce extension E. As specified, we do not think the suggested domain separation between AES-128 and AES-256 improves security. Domain separation would be needed when the same root key is used for several AEAD algorithms. The construction $\mathrm{b0} = E_K(E_K(E) \oplus 1)$ and $\mathrm{b1} = E_K(E_K(E) \oplus 2)$ require sequential AES invocations, which have negative performance implications, and gives $\mathrm{b0} \neq \mathrm{b1}$. An alternative construction could be $\mathrm{b0} = E_K(E)$ and $\mathrm{b1} = E_K(E \oplus C)$, where C is a constant that might have been derived from K.

  XAES-GCM: Uses a 12-byte random value to derive the AEAD key. The 12-byte random nonce N is not used in the key derivation. The use of SP 800-108 reduces the length of the random value compared to directly using AES-CMAC, but defining a general construction is appealing and could be used also for Keccak- and Rijndael-256-based AEADs.

  DNDK-GCM: Uses a 24-byte random value to derive the AEAD key and operates with a fixed nonce. While AES-XGCM and XAES-GCM require 2–3 extra AES invocations per AES-GCM invocation, DNDK-GCM requires 6 extra AES invocations. This is noticeable for short plaintexts but negligible for longer plaintexts. The probability of collisions is the same as in XAES-GCM but higher than AES-XGCM.

  We think a function for deriving key-nonce pairs while keeping AES-GCM unchanged is a valuable idea. While limiting the key length to 256 bits is acceptable, supporting multiple key lengths appears straightforward. Key commitment is useful for specific applications but can be handled through accordion modes.

  None of AES-XGCM, DNDK-GCM, or XAES-GCM use AES-GCM with nonces longer than 96 bits. We think this is the right choice, as it simplifies analysis. Since NIST [1] recommends and IETF [6] mandates restricting the length to 96 bits, many implementations do not support longer nonces.

  The key-nonce derivation function could be defined in a general form to support any NIST-approved AEAD and KDF, or a new custom KDF design providing higher performance. A general function could take as input a root key K, a nonce N, a random nonce extension E, a plaintext P, an associated data A, derive the AEAD key $K' = \mathrm{KDF}(K, E)$, where $|K'| \leq |K|$, and call $\mathrm{AEAD}(K', N, P, A)$. For AES-GCM the length of N should be 96-bits. Excluding the nonce N from the key derivation increases performance.

ExtendedNonceAEAD(K, N, E, P, A):

1. $K' = \mathrm{KDF}(K, E)$
2. $C = \mathrm{AEAD}(K', N, P, A)$
3. Return C

As long as the KDF, AEAD, and the PRG/PRF used to generate K, N, and E are NIST-approved, the ExtendedNonceAEAD construction is already compliant with NIST standards. Any weakness in ExtendedNonceAEAD indicate a weakness or a lack of strict usage limits in a NIST-approved KDF, AEAD, PRG, or PRF that needs to be addressed. We think many NIST-approved KDFs, AEADs, PRGs, and PRFs need stricter usage limits. We analyze ExtendedNonceAEAD under the assumption that the KDF behaves as a theoretical ideal PRF and that any random input is uniformly distributed. The goal is to ensure that the probability of (K', N) collisions remains low.

We note that functions for deriving key-nonce pairs can theoretically address two distinct issues, increasing the number of invocations with a single key K and avoiding the need to maintain state:

1. When used with a deterministic nonce N, the probability of (K', N) collisions is zero. On average, each key K' is used $q \,/\, 2^{|E|}$ times, where q is the number of invocations. For AES-GCM, $|E|$ and q must be chosen so that the probability of encrypting too many blocks for any given key K' is low. For the expected values of $|E|$ and q this will not be a problem.

2. When used with a random nonce N, the construction can be used without state. The probability of a (K', N) collision is $\approx q^2 / 2^{|E|+97}$. An attacker can find a collision by storing (N, E) values in a hash table. The average time complexity is $\approx 2^{|E|+97} / q$ and the memory and data complexities are q. For this attack to have a complicity higher than 2^{λ}, $|E|$ and Q_MAX needs to be chosen so that $|E| + 97 - \log_2 q \geq \lambda$.

In solution 1, only part of N (the counter or the least significant bytes of the counter) needs to be send on the wire, reducing packet overhead. For a detailed discussion on why the complexity of nonce collision attacks must be kept significantly higher than that of distinguishing attacks, see the section "Forbid random nonces" at the end of this document.

A big issue with stateless encryption is that it cannot provide replay protection. SP 800-38C states that a protocol or application should protect against replay attacks, and it has been suggested that NIST should strengthen the recommendations [11]. 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. Systems lacking replay protection are often vulnerable to unexpected attacks and challenging to analyze. If an upper layer was designed with the expectation of replay protection in a lower layer, using a security protocol without replay protection in the lower layers can compromise confidentiality, integrity, and availability in the higher layer, i.e., the whole infosec CIA triad. Serious vulnerabilities due to the lack of replay protection have been common in both standardized and proprietary systems. Stateless encryption is not for general use.

We think NIST should specify a function for deriving key-nonce pairs that can be instantiated with different KDFs and AEADs.

Rijndael-256 variant of GCM:

- The call for comments [2] state that: *"The first approach would be to develop a variant of GCM for a block cipher with 256-bit blocks, such as Rijndael-256, which NIST has proposed for eventual standardization. Such a variant could be a straightforward generalization of the GCM design in which the sizes of the nonce and counter fields are doubled. For example, a limit of 2^{64} invocations would ensure a negligible probability of nonce repetition (about 2^{-64})."*

  The term "counter field" is unclear, as it is not defined in SP 800-38D [1]. While the size of the counter blocks would be doubled to 256 bits, increasing the size of the 32-bit block counter would significantly raise the forgery probability. In GCM, the forgery probability is only $2^{-\tau}$, where $\tau \approx t - \log_2 \ell$, t is the tag length, and $\ell \approx (P\_MAX + A\_MAX) / 16$. A straightforward generalization of AES-GCM with 96-bit nonces would lead to Rijndael-GCM with 224-bit nonces, see [5]. After 2^{64} invocations, the nonce collision probability is $\approx 2^{128} / 2^{225} = 2^{-97}$ and the time complexity of a nonce collision attack is $\approx 2^{64} / 2^{-97} = 2^{161}$, see [12]. We are not sure that a probability of 2^{-64} is negligible, and for practical applications, we think it is preferable to reason about attack complexities and the consequences of attacks. For a detailed discussion on why the complexity of nonce collision attacks must be kept significantly higher than that of distinguishing attacks, see the section "Forbid random nonces" at the end of this document.

- GCM tags do not behave like ideal tags, as the forgery probability depends on the lengths of the plaintext and associated data, as well as prior forgery attempts. During the NIST standardization, Ferguson pointed out two weaknesses in the GCM authentication function [13]. The first weakness significantly increases the probability of successful forgery for long messages. The second weakness reveals the subkey H if an attacker succeeds in creating forgeries. Once H is known, the attacker can consistently forge subsequent messages, drastically increasing the probability of multiple successful forgeries. The two weaknesses are problematic for all tag lengths but are especially critical when short tags are used.

  Galois Counter Mode with Strong Secure Tags (GCM-SST) [5] is a proven secure Authenticated Encryption with Associated Data (AEAD) scheme, building upon and improving GCM to deliver truncated tags with near-ideal forgery resistance, even against multiple forgery attacks. Originally developed by ETSI SAGE, under the name Mac5G, GCM-SST is designed for use in security protocols with replay protection such as TLS, DTLS, QUIC, SRTP, SSH, IPsec, MACsec, and PDCP. In unicast scenarios, the authentication tag behaves like an ideal MAC, including reforgeability resistance. GCM-SST addresses the strong industry demand for fast encryption with minimal overhead and high security. The construction is highly flexible, supporting block ciphers with any block lengths as well as stream ciphers.

  GCM-SST shares most properties with GCM but introduces several key advantages. Unlike GCM, which is restricted to block ciphers with 128-bit block sizes, GCM-SST can be used with any keystream generator. It is designed for security protocols with replay protection, where it provides substantially stronger integrity guarantees. For a given tag length, GCM-SST has strictly better security properties than GCM, and for a fixed security level, GCM-SST has smaller ciphertext expansion. The changes compared to GCM enable truncated tags with replay protection with

near-ideal forgery probabilities, even against multiple forgery attacks. With the parameters in the specification, the forgery probability of GCM-SST is $\approx 1 / 2^t$ and the expected number of forgeries is $\approx v / 2^t$. BSI states that an ideal MAC with a 96-bit tag length is considered acceptable for most applications [14], a requirement that GCM-SST with 96-bit tags satisfies. Achieving a comparable level of security with GCM is nearly impossible. Table 1 provides a comparison of AES-GCM-SST, ChaCha20-Poly1305, and AES-GCM in unicast QUIC.

Name	Tag length (bytes)	Forgery probability before first forgery	Forgery probability after first forgery	Expected number of forgeries
GCM_SST_14	14	$1 / 2^{112}$	$1 / 2^{112}$	$v / 2^{112}$
GCM_SST_12	12	$1 / 2^{96}$	$1 / 2^{96}$	$v / 2^{96}$
POLY1305	16	$1 / 2^{91}$	$1 / 2^{91}$	$v / 2^{91}$
GCM	16	$1 / 2^{116}$	1	$\delta \cdot v^2 / 2^{117}$

Table 1: Comparison between AES-GCM-SST, ChaCha20-Poly1305, and AES-GCM in unicast QUIC where the maximum packet size is 2^{16} bytes. v is the number of decryption queries and δ is the Bernstein bound factor.

The performance of GCM-SST in hardware and software is similar to that of GCM. Compared to AES-GCM implementations where the subkey H is cached between invocations, AES-GCM-SST uses two additional block cipher invocations. This overhead is negligible for most message sizes. In software implementations, it is compensated by using POLYVAL, a "little-endian version" of GHASH, which performs better on little-endian architectures. GCM-SST retains the additive encryption characteristic of GCM, which enables efficient implementations on modern processor architectures.

We think a GCM-like mode of Rijndael-256 would be very useful. Such a mode could securely be used with larger Q_MAX, V_MAX, and nonce length, and offer better confidentiality, see [5]. Due to the weaknesses with GCM, we do not think NIST should specify GCM with any additional block ciphers. For a fixed tag length, GCM-SST has strictly better security properties than GCM, also when used with random nonces and in multicast.

We think NIST should standardize Galois Counter Mode with Strong Secure Tags (GCM-SST) including Rijndael-GCM-SST.

Nonce misuse:

- We agree with NIST's plan to meet the nonce-misuse needs with the development of accordions and derived functions, which will hopefully have excellent security properties. NIST could reconsider if the accordions turn out to be slower than expected.

Stricter confidentiality constraints:

- SP 800-38D requires that $P\_MAX \approx 2^{36}$ but does not define Q_MAX for deterministic nonces. Recent research [15–16] show that distinguishing attacks on counter mode are closely related to plaintext recovery attacks. Consequently, as stated in NIST IR 8459, the key must be changed

well before encrypting $2^{b/2}$ blocks of data, where b is the block size. However, it has been unclear what a reasonable limit is. ANSSI [17] requires rekeying of AES-GCM after 2^{59} blocks, while QUIC [18] requires rekeying after 2^{31} blocks.

As shown in Section 4 of [12], the entropy loss of a pseudorandom permutation (PRP) in counter mode, which provides an upper bound on the amount of information an attacker can theoretically recover, is $\approx \sigma^2 \,/\, 2^b \,/\, \ln 4$ where σ is the number of encrypted blocks. When $\sigma = 2^{64}$, an attacker cannot recover more than $\approx 1 \,/\, \ln 4 \approx 0.721$ bits of information. When $\sigma \leq 2^{59}$, as required by ANSSI, an attacker cannot recover more than $\approx 2^{-10.47} \approx 0.0007$ bits of information. When $\sigma \leq 2^{31}$, as required by QUIC, an attacker cannot recover more than $\approx 2^{-58.47}$ bits of information.

The limits in TLS 1.3, DTLS 1.3, and QUIC seem unnecessarily strict. There is no universally agreed-upon threshold for how much information an attacker must recover for it to constitute a meaningful attack. However, protecting against the recovery of $2^{-58.47}$ bits, a practically negligible fraction, seems overly cautious and unlikely to be a realistic threat.

We agree with the conclusion in NIST IR 8459 that the key must be changed well before encrypting $2^{b/2}$ blocks of data. We recommend NIST aligns with ANSSI [17] by requiring that the maximum number of encrypted blocks under the same key must not exceed $2^{b/2-5}$. This appears to be a well-thought-out and balanced requirement, effectively limiting the amount of information an attacker could recover to no more than ≈ 0.0007 bits. For global industries, alignment between NIST, ANSSI, BSI, Swedish NCSA etc., is crucial. We think NIST should not align with (D)TLS 1.3 and QUIC on this matter.

One reason to have stricter requirements than ANSSI would be in Fully Encrypted Protocols (FEP) [19]. The time complexity of distinguishing attacks is $\approx 2^{b+1} \,/\, \sigma$, and FEPs based on AES might want the complexity of distinguishing attacks to be higher than 2^{70}. However, it's important to note that most practical protocols, including QUIC, are not Fully Encrypted Protocols and do not face the same security concerns.

Due to the narrow 128-bit block size of AES, it is hard to introduce stricter general constraints on Q_MAX or P_MAX without significantly restricting existing applications. Therefore, we suggest that the application utilizing AES-GCM is given the choice of restricting Q_MAX or P_MAX and that the revision of 800-38D adds the following text from [5]:

> "Applications utilizing AES-GCM must enforce limits on P_MAX and/or Q_MAX to ensure that $Q\_MAX \cdot P\_MAX \lessapprox 2^{63}$. This aligns with ANSSI requirements and ensures that an attacker cannot recover more than $\approx 2^{-10.47} \approx 0.0007$ bits of information."

Stricter integrity constraints:

- The authenticity advantage of AES-GCM with 96-bit tags is $\approx \delta \cdot \ell \cdot v \,/\, 2^{127}$ and the expected number of forgeries is $\approx \delta \cdot \ell \cdot v^2 \,/\, 2^{128}$, where v is the number of invocations of the decryption function, $\ell \approx (P\_MAX + A\_MAX) \,/\, 16$, and $\delta \approx 1 + (q+v)^2 \cdot \ell^2 / \, 2^{129}$ is the Bernstein bound factor [20]. SP 800-38D put constraints on ℓ and q, but not v. As a result δ, and thus the integrity advantage and the expected number of forgeries, are unbounded. We think NIST should require that $\delta \approx 1$ to ensure tighter integrity guarantees. Given the narrow 128-bit block size of AES,

introducing stricter general constraints is challenging without significantly restricting existing applications. Therefore, we suggest that application utilizing AES-GCM is given the choice of restricting P_MAX, A_MAX, Q_MAX, and/or V_MAX and that the revision of 800-38D adds the following text from [5]:

> "Applications utilizing AES-GCM must enforce limits on P_MAX, A_MAX, Q_MAX, and/or V_MAX to ensure that $(P\_MAX + A\_MAX) \cdot (Q\_MAX + V\_MAX) \lessapprox 2^{66}$. This ensures that $\delta \approx 1$."

Forbid random nonces with a fixed key:

- Theoretical cryptographers often treat all attacks affecting confidentiality as equivalent. However, this approach can be impractical for real-world applications. At best, it reduces efficiency by enforcing unnecessary high security against some attack types. At worst, it undermines security by assuming that no attack on confidentiality requires a higher complexity than distinguishing attacks. This assumption is particularly dangerous for block cipher modes with a narrow 128-bit block size, where the security against distinguishing attacks is relatively low.

  As discussed in [12], the time complexity of nonce collision attacks when AES-GCM is used with random nonces is $\approx \min(2^{|IV|+1}, 2^{129}) / q$. The complexity of distinguishing attacks is $\approx 2^{129} / q$. While it should be clear that random 96-bit nonces do not provide sufficient security and should be forbidden, some argue that using longer random nonces is acceptable because the attack complexity is similar to that of distinguishing attacks. We strongly disagree with this view.

  If we compare the severity of attacks using the lack of collisions (distinguishing attacks) with attacks using collisions (collision attacks) we see that the amount of information an attacker can recover is vastly different. Assuming AES-GCM with $|IV| = 128$, plaintexts with length $\approx 2^{36}$ bytes and $q = 2^{27}$, which follows ANSSI's requirements, the complexity of both collision and distinguishing attacks are $\approx 2^{102}$. While a single nonce collision attack can recover up to 2^{40} bits of plaintext in addition to enabling universal forgery, the entropy loss of a pseudorandom permutation (PRP) in counter mode, which provides an upper bound on the information an attacker can theoretically recover, is $\approx \sigma^2 \,/\, 2^b \,/\, \ln 4$, where σ is the number of encrypted blocks [12]. In a distinguish attack, the attacker can therefore not recover more than $\approx 2^{-10.47}$ bits of information, i.e., $\approx 2^{50.5}$ times less than in the nonce collision attack.

  Previously, we recommended that NIST discourage the use of AES-GCM with random nonces [4]. Given the devastating consequences of collision attacks, the need to address them independently of distinguishing attacks, and the practicality of deriving random key-nonce pairs for all stateless encryption scenarios, we now recommend that NIST forbid the use of random nonces with a fixed key. Instead of allowing random nonces, NIST should define a method for deriving random key-nonce pairs.

Guidance on random nonces:

- Rogaway states Section 12.4.10 of [21] that:

  > *"the exposition in the NIST spec seems to kind of "fall apart" in Sections 8 and 9, and in Appendix C. These sections stray from the goal of defining GCM, and make multiple*

incorrect or inscrutable statements. Here are some examples. Page 18 : ***The probability that the authenticated encryption function ever will be invoked with the same IV and the same key on two (or more) sets of input data shall be no greater than*** 2^{-32} *(here and later in this paragraph, imperatives are preserved in their original bold font). The probabilistic demand excludes use of almost all cryptographic PRGs (including those standardized by NIST), where no such guarantee is known."*

Theoretically, using a cryptographic pseudorandom generator (PRG) for generating a large number of non-colliding IVs seems like the wrong approach. Instead, a pseudorandom function family (PRF) should be utilized. While a PRG ensures that a single output appears random, a PRF guarantees that all outputs appear random.

NIST should give examples of PRGs or PRFs that are approved for generating a large number of non-colliding random nonces.

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

References

[1] NIST SP 800-38D, GCM and GMAC Block Cipher Modes of Operation
https://csrc.nist.gov/pubs/sp/800/38/d/final

[2] Pre-Draft Call for Comments: GCM and GMAC Block Cipher Modes of Operation
https://csrc.nist.gov/pubs/sp/800/38/d/r1/iprd

[3] Ericsson's Comments on SP 800-38D
https://csrc.nist.gov/csrc/media/Projects/crypto-publication-review-project/documents/initial-comments/sp800-38d-initial-public-comments-2021.pdf

[4] Ericsson's Comments on SP 800-38D Decision Proposal
https://csrc.nist.gov/csrc/media/Projects/crypto-publication-review-project/documents/decision-proposal-comments/sp800-38d-decision-proposal-comments-2023.pdf

[5] Galois Counter Mode with Strong Secure Tags (GCM-SST)
https://datatracker.ietf.org/doc/html/draft-mattsson-cfrg-aes-gcm-sst

[6] RFC 5116, An Interface and Algorithms for Authenticated Encryption
https://www.rfc-editor.org/rfc/rfc5116.html

[7] Introducing AES-XGCM
https://soatok.blog/2022/12/21/extending-the-aes-gcm-nonce-without-nightmare-fuel/

[8] The XAES-256-GCM extended-nonce AEAD
https://c2sp.org/XAES-256-GCM

[9] Double Nonce Derive Key AES-GCM (DNDK-GCM)
https://datatracker.ietf.org/doc/html/draft-gueron-cfrg-dndkgcm-00

[10] Double Nonce Derive Key AES-GCM (DNDK-GCM)
https://csrc.nist.gov/csrc/media/Presentations/2024/double-nonce-derive-key-gcm-dndk-gcm/images-media/sess-6-gueron-acm-workshop-2024.pdf

[11] Ericsson's Comments on SP 800-38C
https://csrc.nist.gov/csrc/media/Projects/crypto-publication-review-project/documents/initial-comments/sp800-38c-initial-public-comments-2024.pdf

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

[13] Authentication weaknesses in GCM
https://csrc.nist.gov/CSRC/media/Projects/Block-Cipher-Techniques/documents/BCM/Comments/CWC-GCM/Ferguson2.pdf

[14] BSI TR-02102-1, Cryptographic Mechanisms: Recommendations and Key Lengths
https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.html

[15] Impossible plaintext cryptanalysis and probable-plaintext collision attacks of 64-bit block cipher modes
https://eprint.iacr.org/2012/623.pdf

[16] The Missing Difference Problem, and Its Applications to Counter Mode Encryption
https://doi.org/10.1007/978-3-319-78375-8_24

[17] ANSSI PG-083, Guide des mécanismes cryptographiques
https://cyber.gouv.fr/sites/default/files/2021/03/anssi-guide-mecanismes_crypto-2.04.pdf

[18] RFC 9001, Using TLS to Secure QUIC
https://www.rfc-editor.org/rfc/rfc9001.html

[19] Security Notions for Fully Encrypted Protocols
https://www.petsymposium.org/foci/2023/foci-2023-0004.pdf

[20] Breaking and Repairing GCM Security Proofs
https://eprint.iacr.org/2012/438.pdf

[21] Evaluation of Some Blockcipher Modes of Operation
https://web.cs.ucdavis.edu/~rogaway/papers/modes.pdf