

Second Pre-Draft Public Comments on Proposed Revision of SP 800-38D

Comment period: June 1, 2026 – July 31, 2026

NIST’s cipher modes team [issued a second pre-draft call for public comments](#) for the planned revision of Special Publication (SP) 800-38D, *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*, 2007. The comments that NIST received during the comment period are collected below. Information about NIST’s block cipher modes of operation is available at the [project site](#), and ongoing discussion of technical questions may be found at the [NIST cipher modes forum](#).

LIST OF COMMENTS

1.	Legacy Roots Housing Initiative (Sloane Ebonie King-Wolf), June 1, 2026.....	2
2.	QUBITRONIX (Wasif Khan), June 10, 2026.....	3
3.	Anitha Dakamarri, July 4, 2026.....	8
4.	Microsoft Corporation (Mark Svancarek), July 7, 2026.....	11
5.	Ericsson (John Preuß Mattsson), July 10, 2026.....	15
6.	Additional Comments from LRHI (Sloan “Chyna” King-Wolf), July 22, 2026	23
7.	Chunghwa Telecom Laboratories (Abel C. H. Chen), July 23, 2026	24
8.	Niranjan Sharma, July 26, 2026.....	26
9.	Revised Comments from Atsec (Yi Mao), July 27, 2026.....	28
10.	Jean Paul Degabriele, Jan Gilcher, Jérôme Govinden, Kenneth G. Patterson, July 29, 2026	31
11.	Roberto Avanzi, July 30, 2026	37
12.	Abhishek Chowdhury, July 30, 2026.....	38
13.	Akiko Inoue, Tetsu Iwata, Kazuhiko Minematsu, July 31, 2026.....	42
14.	Thales (Graham COSTA), July 31, 2026.....	55
15.	Google (Victor Escobedo), July 31, 2026.....	57
16.	NSA Center for Cybersecurity Standards (Denisha Warren), July 31, 2026.....	59

1. Legacy Roots Housing Initiative (Sloane Ebonie King-Wolf), June 1, 2026

Comments on SP 800-38D revision

To the NIST Cipher Modes Project Team, 

As the Founder of Legacy Roots Housing Initiative (LRHI), LLC, an organization committed to integrating high-security digital financial infrastructure into modular construction and community housing, I submit the following feedback regarding the revision of SP 800-38D and the proposed wGCM variant.

General Feedback:

The integration of wGCM to support Rijndael-256 is a significant step toward modernizing cryptographic standards. As our operational model relies on AI-supported operational continuity and secure data strategies, robust and efficient block cipher modes are critical for our infrastructure.

Specific Considerations:

- **GCM Authentication Tags:** Regarding the proposed removal of tags less than 96 bits, I support the limitation to 96-bit and 128-bit tag sizes as sufficient for modern secure applications.
- **wGCM Hash Function (wide-GHASH vs. concat-GHASH):** From an operational and security perspective, I lean toward the wide-GHASH option. While concat-GHASH may offer efficiency gains, the superior generic security assurance provided by wide-GHASH is preferable for systems that manage long-term community infrastructure data and federal project documentation. Ensuring the highest level of security assurance is paramount for our adoption of new standards.
- **Operational Constraints:** The proposed 264 limit for wGCM invocations per key appears sufficient for current and projected scaling of our operational continuity systems.

Thank you for your work in maintaining these vital standards. We appreciate the opportunity to contribute to the security of the broader cryptographic ecosystem.

Sincerely,

Sloan Ebonie King-Wolf

Founder | Legacy Roots Housing Initiative (LRHI), LL

2. QUBITRONIX (Wasif Khan), June 10, 2026

Please see attached my formal submission for requested comments. Let me know if you have any further questions.

Warm Regards,



Wasif Khan

CEO

Comments on NIST SP 800-38D Revision 1 (Second Preliminary Draft)

Second Pre-Draft Call for Comments: GCM and GMAC Block Cipher Modes of Operation

Submitted To: National Institute of Standards and Technology, Cryptographic Technology Group
(ciphermodes@nist.gov)
Subject: Comments on SP 800-38D revision
Submitted by: Wasif Khan [email address]
Date: June 10, 2026

I am very excited and really appreciate the opportunity to comment on the second pre-draft call for comments on the revision of SP 800-38D that was announced on June 1, 2026. I have tried my best to provide the responses below to address each of the questions posed by NIST concerning the revised Galois/Counter Mode (GCM) and the proposed wide variant (wGCM) intended to operate with 256-bit block ciphers such as Rijndael-256.

Executive Summary:

The threat landscape now includes highly capable automated and AI-assisted vulnerability discovery operating at fleet scale. These systems routinely identify and exploit configuration errors, boundary-condition violations, and nonce-management failures—including those introduced through normal implementation and operational mistakes. In this setting, designs should emphasize simplicity, larger safety margins, and constraints that are difficult to violate unintentionally, even if that comes with some efficiency cost.

Accordingly, I recommend reducing optionality and widening margins: limit each mode to two tag sizes; use fixed-length IVs without hash-derived derivation; require full-entropy, non-repeating nonces; standardize on a single deterministic IV construction; and prefer wide-GHASH over more efficient but more tightly bounded alternatives. These choices are intended to reduce the likelihood of misuse in real-world deployments. At fleet scale, eliminating edge cases and increasing tolerance for mistakes provides more practical security value than marginal performance gains.

1. Comments on GCM

1.1. Tag sizes (Question 1)

Recommendation: Tag sizes of 96 and 128 bits are sufficient. NIST should remove the intermediate sizes of 104, 112, and 120 bits, designate **128 bits as the default**, and retain *96 bits only as an interoperability floor* for constrained or legacy protocol profiles (for example, certain IPsec and embedded deployments).

Rationale: The intermediate sizes see very limited deployment in practice; most protocols use either the full 128-bit tag or 96 bits in constrained environments. Each additional truncation option multiplies the negotiation and configuration surface that automated misconfiguration discovery can probe, while adding negligible security value. Because the forgery advantage for truncated GCM tags degrades with message length, removing sub-96-bit tags is clearly correct, and consolidating on two sizes simplifies validation and conformance testing. A brief survey of deployed systems is advisable before final removal in case a legacy profile depends on an intermediate size.

1.2. Variable-length IVs (Question 2)

Recommendation: Variable-length IVs are not important for modern applications. **NIST should deprecate the GHASH-based derivation** of the pre-counter block from non-96-bit IVs and standardize on 96-bit IVs, *accompanied by clear migration guidance.*

Rationale: The variable-length IV path is a long-recognized source of fragility: it admits IV collisions through hash collisions, complicates implementations, and is rarely exercised because the overwhelming majority of deployments use 96-bit IVs. Removing it shrinks the implementation and eliminates a subtle corner case that is difficult to implement correctly and easy for automated analysis to probe. A small number of legacy systems use non-96-bit identifiers as IVs; deprecation with migration guidance, rather than immediate removal, accommodates them.

2. 2. Comments on wGCM

2.1. Invocation limit (Question 1)

Recommendation: Yes, a **limit of 2^{64} invocations per key is sufficient.**

Rationale: With 256-bit blocks, the relevant birthday bound sits near 2^{128} blocks, so 2^{64} invocations leaves an enormous margin and far exceeds GCM's effective ceiling of about 2^{32} invocations under random IVs. NIST should, however, *pair the invocation limit with an explicit total-data limit per key*, so that high-throughput systems cannot erode the security margin through very large individual messages while remaining within the invocation count.

2.2. Authentication tag sizes (Question 2)

Recommendation: **Specify two tag sizes: 128 and 256 bits.** More than one size is necessary; 192 bits should be omitted.

Rationale: A 128-bit tag serves as the efficient floor for high-throughput links where per-record overhead matters. A 256-bit tag preserves the forgery margin at the large message and data limits that wGCM exists to unlock, since the authentication bound degrades with message length in blocks. The specification should tie its largest permitted data limits to the 256-bit tag. *A 192-bit option adds negotiation surface without a compelling use case.*

2.3. Longer IVs via the internal hash function (Question 3)

Recommendation: **No. NIST should keep the fixed 192-bit IV** with the remaining 64 bits of the IV block devoted to the message block counter, and should not support hash-derived variable-length IVs.

Rationale: A 192-bit IV space is ample for both deterministic and random constructions, and a 64-bit block counter permits messages of up to 2^{64} 32-byte blocks (approximately 2^{69} bytes), far beyond any realistic message. Importing GCM's hash-derived IV mechanism would *carry that mode's most error-prone feature into an otherwise clean design and would reintroduce an entire class of parsing and collision pitfalls.*

2.4. Deterministic IV construction (Question 4)

Recommendation: The proposed split of a 96-bit fixed field and a 96-bit invocation field is sufficient. **No additional options are needed.**

Rationale: *A 96-bit invocation field cannot wrap before the 2^{64} invocation limit forces rekeying*, and 96 bits of fixed field accommodates rich device and context namespaces. A single canonical layout is itself a security property: interoperability around one well-known format means fewer bespoke implementations in which boundary and parsing bugs can hide. Permitting alternative splits *would invite configuration errors for little practical benefit.*

2.5. Free field in the random IV construction (Question 5)

Recommendation: **The free field is neither useful nor necessary.** The full 192 bits should be required to be random.

Rationale: With 192 random bits and at most 2^{64} invocations per key, the IV collision probability is bounded by approximately 2^{-64} , which is acceptable for all foreseeable applications. Any free field can *only take entropy from the random field and raise the collision probability*, and an **IV collision in the GCM family is catastrophic**: it can expose the hash subkey and enable forgeries. Applications that need to bind contextual information should carry it in the additional authenticated data rather than inside the IV.

2.6. Flexibility of the two IV constructions (Question 6)

Recommendation: The deterministic and random constructions are sufficiently flexible for this specification. **No additional constructions should be added** to this document.

Rationale: The two constructions cover the standard stateful and stateless deployment models, and a *minimal set simplifies validation and conformance testing*.

One related item merits NIST's attention outside this specification: Because nonce reuse is the highest-consequence failure mode in this family, and precisely the class of defect that automated fleet-wide analysis is now effective at discovering, **NIST should flag a nonce-misuse-resistant (SIV-style) construction as a candidate for a companion specification**.

2.7. Hash function: wide-GHASH versus concat-GHASH (Question 7)

Recommendation: **NIST should specify wide-GHASH**, the hash function defined by a suitable polynomial of degree 256 over $GF(2^{256})$, *as the hash component of wGCM*.

Rationale: wGCM's purpose is to support larger data, message, and invocation limits than GCM. concat-GHASH would achieve better performance on existing 128-bit carryless-multiplication hardware, *but only at the cost of materially weaker generic security assurance and substantially more restrictive usage bounds, which would undermine the principal motivation for the new mode*. Security that holds only beneath tighter, easier-to-exceed limits is exactly the failure mode that automated analysis detects and exploits. If concrete performance data later shows a compelling need, concat-GHASH could be offered as a clearly labeled optional profile with conservative, explicitly documented usage limits, **but it should not be the default and should not be required for conformance**.

3. Summary of recommendations

Question	Recommendation
GCM Q1: Tag sizes	Retain 96 and 128 bits only; 128 as default. Remove 104, 112, and 120.
GCM Q2: Variable-length IVs	Deprecate; standardize on 96-bit IVs with migration guidance.
wGCM Q1: Invocation limit	2^{64} per key is sufficient; couple with an explicit total-data limit.
wGCM Q2: Tag sizes	Specify 128 and 256 bits; omit 192. Tie largest data limits to the 256-bit tag.
wGCM Q3: Longer IVs via hashing	No. Keep the fixed 192-bit IV with 64-bit block counter.

wGCM Q4: Deterministic IV split	The single 96/96 split is sufficient; no alternatives needed.
wGCM Q5: Free field	Unnecessary; require the full 192 bits to be random.
wGCM Q6: Construction flexibility	Two constructions suffice; flag SIV-style misuse resistance for a companion specification.
wGCM Q7: Hash function	Specify wide-GHASH as primary; concat-GHASH at most as a clearly bounded optional profile.

In summary, t

Thank you for your consideration of these comments.

Respectfully submitted,

Wasif

[email address]

3. Anitha Dakamarri, July 4, 2026

Hello NIST team,

I have been drafting my feedback from last few weeks and please find below are the review comments!

1. Questions on GCM:

- a. The revision to GCM will remove support for authentication tags whose lengths are less than 96 bits. Which of the remaining tag sizes (i.e., 96, 104, 112, 120, or 128 bits) are needed for GCM? In particular, would tag sizes of 96 bits and 128 bits be sufficient for applications?

My comments: I support the decision to remove support for authentication tag lengths below 96 bits. Given current threat models and long-term security requirements, I believe 96 bit tags should be considered the practical minimum.

From my perspective, supporting both 96 bit and 128 bit tag lengths would likely satisfy the majority of real world applications. A 96 bit tag provides a good balance between bandwidth efficiency and security for constrained environments, while a 128 bit tag offers stronger security margins for high-value and long lived deployments.

I don't see a strong need for intermediate tag lengths (104, 112, and 120 bits) unless there are specific legacy interoperability requirements. Simplifying implementation choices by focusing on 96 bit and 128 bit tags could reduce complexity while maintaining broad applicability.

- b. GCM currently supports variable-length IVs by applying the internal hash function to generate a new, 96-bit IV value. Are variable-length IVs important for any applications?

My comments: I believe support for variable length IVs remains valuable for interoperability and compatibility with existing deployments. Many applications, protocols, and hardware implementations have already adopted non-standard IV formats, and removing this capability could create migration challenges.

However, I also recognize that fixed-length 96-bit IVs provide significant implementation and performance advantages and reduce the likelihood of IV construction errors. Therefore, I would recommend retaining support for variable-length IVs while continuing to emphasize 96 bit IVs as the preferred and recommended deployment option.

2. Questions on wGCM:

Would a limit of 2^{64} wGCM invocations per key be sufficient?

My comments: I think the limit of 2^{64} should be more than sufficient for practical deployments. This limit represents a substantial increase over current GCM usage bounds and appears appropriate given the larger block size and expected use cases for Rijndael 256 based systems.

For most operational environments, cryptographic keys would be rotated well before reaching such limits due to organizational security policies, compliance requirements, or lifecycle management practices.

What authentication tag sizes should be specified for wGCM (e.g., 128, 192, or 256 bits) to support larger data limits than GCM? Is more than one tag size necessary?

My comments: May be im thinking too much but I feel like 128 bit and 256 bit authentication tags are likely sufficient for most use cases.

A 128 bit tag provides strong security while maintaining compatibility with existing security expectations established by GCM. A 256 bit tag would be beneficial for environments seeking security parity with 256 bit cryptographic primitives and for long-term protection of highly sensitive data.

I am less convinced that a 192 bit tag provides significant practical benefit compared to offering only 128 bit and 256 bit options. Reducing the number of supported tag lengths could improve simplicity and interoperability. (Ignore this comment if it my opinion is amateur)

NIST proposes 192 bits as the default IV length for wGCM, devoting the remaining 64 bits of the IV block within wGCM to the message block counter for the GCTR encryption. Is there a need to support longer IVs in wGCM by applying the internal hash function, as in GCM?

My comments: I believe the proposed default IV length of 192 bits is appropriate and should satisfy most use cases.

But, can I suggest supporting longer IVs through the same hash based processing approach currently used in GCM. Maintain this flexibility would improve interoperability and accommodate specialized deployments that may require non-standard IV formats without introducing significant implementation burden.

The deterministic IV construction for GCM defines a fixed field and an invocation field. For wGCM, NIST proposes to recommend a fixed field of 96 bits and an invocation field of 96 bits to promote interoperability. Are other options needed?

My comments: The proposed deterministic IV construction consisting of a 96 bit fixed field and 96 bit invocation field appears reasonable and promotes interoperability across implementations. from my limited perspective, standardizing on a single deterministic construction will reduce implementation errors and improve consistency. Unless there are well established use cases that require alternative structures, I like to prefer to avoid introducing additional deterministic IV formats..

The random IV construction for GCM defines a random field and a free field. For wGCM, if the number of invocations per key were limited to 2^{64} , and if 192 bits were specified for the random field, then the probability of an IV collision would be limited to 2^{-64} . Is the free field useful/necessary for wGCM?

My comments: If 192 bits are dedicated to randomness and invocation limits remain bounded at 2^{64} , I do not see a strong security requirement for an additional free field. In my limited experience, additional flexibility fields frequently increase implementation complexity and the risk of inconsistent usage. Unless a clear operational requirement is identified, I like to suggest only keeping the IV construction simple and allocating the entire field to randomness.

Are these two 192-bit IV constructions sufficiently flexible, or should NIST consider adding other, comparably secure constructions?

My comments: I believe the two proposed 192-bit IV constructions provide sufficient flexibility for most deployment scenarios.

One deterministic option and one randomized option generally cover the majority of operational requirements. Introducing additional constructions may increase complexity, reduce interoperability, and create new opportunities for implementation mistakes.

Hope the above feedback is helpful, if not kindly ignore.

Little intro about me as below:

I am Anitha Dakamarri, a seasoned IT professional with over 19 years of experience in the field of information security. Over the years, I have honed my skills in various domains, including threat modeling, application security assessments, and network security assessments. My work experience spans across reputable organizations like DFIN, USCELLULAR, and Syntel Inc., where I have led teams, developed security standards, and implemented robust security frameworks. hold several industry certifications like CISSP,CISM,CEH,CHFI, which reflect my commitment to continuous learning and professional growth. My passion for security extends beyond technical assessments; Im active speaker and volunteer at ISACA(CISM Exam development, Mentor), ISC2(Exam development and UBK panel, 50*50 volunteer) OWASP(Contributor, CFP review), Women4Cyber (Mentor) and WiCys(Mentor), CSA(STAR Contributor)!

Thanks,
Anitha D.

4. Microsoft Corporation (Mark Svancarek), July 7, 2026

NIST colleagues:

Attached are Microsoft's comments on the Second Pre-Draft Call for Comments on the revision of NIST Special Publication 800-38D, *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*, including our cover letter and completed comment submission.

We appreciate the opportunity to provide feedback and welcome NIST's continued work to evolve and clarify guidance for GCM and GMAC, including consideration of parameters for wGCM.

If you have any questions or would like to discuss our submission further, please feel free to contact us.

Best regards,

Mark Svancarek
NIST Program Management Office



Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

Tel 425 882 8080
Fax 425 706 7329
www.microsoft.com

June 30, 2026

National Institute of Standards and Technology
Cybersecurity and Privacy Program
100 Bureau Drive, Stop 1070
Gaithersburg, MD 20899

Microsoft Comments on SP 800-38D revision — GCM and GMAC Block Cipher Modes of Operation

Dear NIST Cybersecurity and Privacy Program:

Microsoft appreciates the opportunity to provide comments in response to NIST's Second Pre-Draft Call for Comments on the revision of Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. Microsoft supports NIST's continued work to modernize the guidance for GCM and GMAC and to evaluate parameters for a wider variant, wGCM, operating on 256-bit blocks.

Our comments focus on simplifying the revised recommendations where possible, preserving interoperability where necessary, and selecting clear security parameters for new protocols, applications, and APIs. We encourage NIST to prefer strong defaults, fixed and unambiguous nonce constructions, and implementation guidance that minimizes avoidable interoperability risk.

Microsoft's responses to NIST's specific questions are provided below in the body of this letter.

Requested Feedback

1. GCM (Questions 1a–1b)

NIST Request	Microsoft Response
1(a) The revision to GCM will remove support for authentication tags whose lengths are less than 96 bits. Which of the remaining tag sizes (96, 104, 112, 120, or 128 bits) are needed? Would 96-bit and 128-bit tags be sufficient?	Microsoft observes that 128-bit authentication tags are in broad use with GCM today and supports deprecating tag sizes below 128 bits for new protocols, applications, and APIs. If there is a compelling need for a shorter tag size, limiting support to 96-bit and 128-bit tags would be preferable to retaining all byte-sized options. Existing cryptographic APIs may continue to support currently available tag sizes for backward compatibility and interoperability.

1(b) GCM currently supports variable-length IVs by applying the internal hash function to generate a new, 96-bit IV value. Are variable-length IVs important for any applications?	Microsoft does not consider variable-length IVs necessary for modern use. Core Microsoft cryptographic libraries have historically used fixed 96-bit nonces, and Windows APIs continue to support 96-bit nonces. Variable-length nonce support exists primarily for interoperability, and the standard would be simpler and stronger without this option.
--	--

2. wGCM (Questions 2a–2g)

NIST Request	Microsoft Response
2(a) Would a limit of 2^{64} wGCM invocations per key be sufficient?	Microsoft considers a limit of 2^{64} wGCM invocations per key reasonable and sufficient.
2(b) What authentication tag sizes should be specified for wGCM (e.g., 128, 192, or 256 bits)? Is more than one tag size necessary?	Microsoft recommends specifying a single 256-bit authentication tag for wGCM, consistent with a strong default and avoiding unnecessary variation.
2(c) Is there a need to support longer IVs in wGCM by applying the internal hash function, as in GCM?	Microsoft recommends supporting only the nonce length that reflects the underlying construction. A fixed 192-bit nonce for wGCM makes sense, and Microsoft strongly recommends against supporting longer or shorter nonces that are internally hashed.
2(d) Are other deterministic IV constructions needed beyond the proposed fixed and invocation fields?	Microsoft recommends using a 128-bit fixed field and a 64-bit invocation field. Because wGCM invocations are already limited to 2^{64}, a larger invocation field appears to add little value and may create interoperability risk if implementations handle 64-bit and 96-bit increments differently. Microsoft also recommends removing the LFSR option, using an integer counter modulo 2^{64}, and explicitly specifying endianness. Overall, a single clear option is preferable to a menu of options.
2(e) Is the free field useful/necessary for wGCM random IV construction?	If wGCM includes a random nonce construction, Microsoft recommends using the full 192-bit nonce as random input with no free field.
2(f) Are the proposed IV constructions sufficiently flexible?	Microsoft considers the proposed wGCM IV constructions sufficient and recommends considering single-use randomly generated wGCM keys over a random nonce construction where practical.



Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

Tel 425 882 8080
Fax 425 706 7329
www.microsoft.com

2(g) Should NIST consider alternative hash constructions such as concat-GHASH?	Microsoft recommends using a 256-bit polynomial for wide-GHASH and strongly recommends specifying bit endianness in a way that avoids unnecessary byte shuffling of ciphertexts. In software, concat-GHASH does not appear sufficiently more efficient than a 256-bit wGHASH polynomial to justify the added security complexity. Both wider GHASH approaches require roughly twice the raw multiplication work of existing GCM, and optimizing the modular reduction step at the cost of security complexity is not recommended.

Conclusion

Microsoft appreciates NIST's leadership in maintaining and strengthening cryptographic standards that support secure and interoperable implementations. We encourage NIST to continue prioritizing clarity, minimal parameter variation, and implementation guidance that enables broad, consistent adoption.

We look forward to continued engagement as NIST develops the revision to SP 800-38D and the associated guidance for wGCM.

Respectfully submitted,

Mark Svancarek
NIST Program Management Office
Corporate Standards Group, Microsoft Corporation

5. Ericsson (John Preuß Mattsson), July 10, 2026

Dear NIST,

Thanks for your continuous efforts to produce well-written open-access security documents. Please find attached our comments on the Second Pre-Draft Call for Comments on NIST SP 800-38D.

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

Ericsson AB
Group Function Technology
SE-164 80 Stockholm
SWEDEN

Comments on NIST SP 800-38D (GCM, GMAC, and wGCM)

Dear NIST,

Thanks for your continued efforts to produce well-written, user-friendly, and open-access security documents. We welcome NIST's plans to revise SP 800-38D and to standardize Rijndael-256 and wGCM. 3GPP and ETSI SAGE have already standardized Rijndael-256 for use in the MILENAGE-256 authentication and key-generation function [1]. Below, we provide feedback on the questions raised in [2], building on Ericsson's previous comments in [3–6].

In the comments below, we use the following notation:

- q , the number of encryption queries
- v , the number of decryption queries
- ℓ , the maximum number of authenticated blocks (plaintext + associated data)
- ℓ_p , the maximum number of plaintext blocks
- η , the length in bits of the random field in the RBG-based IV construction
- τ , the authentication tag length in bits

Questions on GCM:

- *"The revision to GCM will remove support for authentication tags whose lengths are less than 96 bits. Which of the remaining tag sizes (i.e., 96, 104, 112, 120, or 128 bits) are needed for GCM? In particular, would tag sizes of 96 bits and 128 bits be sufficient for applications?"*

We are not aware of any applications using 104-, 112-, or 120-bit GCM tags. While 96- and 128-bit tags appear sufficient, we see no compelling reason to restrict the current 96–128-bit range. However, NIST should significantly reduce the maximum length of the associated data A . With $\ell = 2^{57}$ blocks of authenticated data, the forgery probability for 96-bit tags is $\approx 2^{-39}$, which is clearly not what users expect. NIST should also restrict the number of invocations for deterministic 96-bit IVs to ensure that the Bernstein bound factor [7]



$$\delta_{128} \lesssim 1 + (q\ell_p + v)^2 / 2^{129} ,$$

remains close to 1. The Bernstein bound factor δ depends on the total number of block cipher invocations σ , which we for simplicity upper-bound by $\sigma \lesssim q \cdot \ell_p + v$. Furthermore, NIST should provide guidance on the enforcement of invocation limits in group-key environments, including one-to-many, many-to-one, and many-to-many communication scenarios.

New applications where byte efficiency and strong security matter should follow the example of 5G and 6G systems and adopt GCM with Strong Secure Tags (GCM-SST) [8], designed by ETSI SAGE and standardized by 3GPP in TS 35.240–35.248. GCM-SST mitigates several known vulnerabilities in GCM, significantly reducing integrity advantage bounds and the expected number of successful forgeries.

- *“GCM currently supports variable-length IVs by applying the internal hash function to generate a new, 96-bit IV value. Are variable-length IVs important for any applications?”*

Longer IVs need to be discussed together with random IVs and Nonce-based Key Derivation (NKD). The observation that *“when the nonce length is restricted to 96 bits, GCM has better security bounds”* [3, 7] applies only to deterministic IVs. Random IVs shorter than the block size degrade both the privacy and authenticity bounds of GCM [7, 9] by introducing an additional $q^2 / 2^{\eta+1}$ term. NIST should not prohibit long IVs while continuing to permit random IVs. Nonce-based Key Derivation (NKD) appears to be a superior approach for addressing the same use case as random IVs, and NKD is already deployed in several large-scale systems [10–11]. We suggest that NIST mandate deterministic 96-bit IVs and specify NKD for all NIST-approved encryption modes. We note that NKD, when constructed by combining an approved KDF with an approved AEAD, is already approved, but would benefit from additional guidance.

Questions on wGCM:

- *“Would a limit of 2^{64} wGCM invocations per key be sufficient?”*

A limit of 2^{64} invocations would be sufficient for all use cases. However, 2^{64} invocations cannot be combined with message lengths of 2^{64} blocks without incurring security degradation due to the Bernstein bound factor [7]

$$\delta_{256} \lesssim 1 + (q\ell_p + v)^2 / 2^{257} .$$

To maintain $\delta_{256} \approx 1$, which should be a strict requirement, one option is to require that

$$q\ell_p + v \ll 2^{128} .$$

Furthermore, a single key should not be used for such a large number of invocations. To align with zero-trust principles and minimize the impact of key compromise, rekeying should occur well before reaching 2^{64} invocations. Even in historical systems where keys were distributed on paper, high-security practice commonly employed a cryptoperiod of one day [12]. Modern guidance recommends rekeying via ephemeral asymmetric key exchange providing Forward Secrecy (FS) and Post-Compromise Security (PCS) after 1 hour or 2^{25} – 2^{32} 256-bit blocks [13–15].



While we believe NIST should increase the invocation and plaintext-length limits relative to GCM, we are cautious about increasing the limit on the total number of blocks encrypted under a single key. Even if cryptographic bounds permit larger volumes of encrypted data, NIST should recommend much stricter limits to reduce the consequences of key compromise.

- *"What authentication tag sizes should be specified for wGCM (e.g., 128, 192, or 256 bits) to support larger data limits than GCM? Is more than one tag size necessary?"*

Rather than asking what tag lengths should be specified, the relevant question is what level of security the tags should provide. Let $E(F)$ denote the expected number of successful forgeries. Assuming a sufficiently large key size such that brute-force key-recovery attacks can be neglected, a strong integrity mechanism should satisfy

$$E(F) \approx v/2^\tau ,$$

for all allowed values of q , v , ℓ , and τ . We recommend that all future NIST standards, including wGCM, target this robust security definition. Users, developers, and standards authors typically expect that $E(F) \approx v/2^\tau$.

We believe $E(F) \approx v/2^{128}$ provides sufficient security for all use cases, as it likely falls below the probability of a verification incorrectly succeeding due to environmental phenomena, such as cosmic rays and thermal noise. Shorter tags with $E(F) \approx v/2^\tau$ are, however, needed in many use cases. The fact that both IETF [16] and 3GPP [8] have recently standardized new AEADs demonstrates that NIST's existing AEADs do not meet all industry requirements.

- *"NIST proposes 192 bits as the default IV length for wGCM, devoting the remaining 64 bits of the IV block within wGCM to the message block counter for the GCTR encryption. Is there a need to support longer IVs in wGCM by applying the internal hash function, as in GCM?"*

We believe wGCM should support a single fixed IV length smaller than the block size. A 192-bit IV appears appropriate. However, message lengths of 2^{64} blocks cannot be combined with 2^{64} invocations due to the birthday bound and zero-trust principles; see our answer on the invocation limit.

- *"The deterministic IV construction for GCM defines a fixed field and an invocation field. For wGCM, NIST proposes to recommend a fixed field of 96 bits and an invocation field of 96 bits to promote interoperability. Are other options needed?"*

The length of the invocation field should be aligned with the invocation limit and CPU-efficient data sizes. A 128-bit fixed field may be useful for larger identities and contexts. We suggest NIST recommend a 128-bit fixed field and a 64-bit invocation field.

- *"The random IV construction for GCM defines a random field and a free field. For wGCM, if the number of invocations per key were limited to 2^{64} , and if 192 bits were specified for the random field, then the probability of an IV collision would be limited to 2^{-64} . Is the free field useful/necessary for wGCM?"*



Rather than discussing collision probabilities, NIST should discuss security strength. Random IVs shorter than the block size degrade both the privacy and authenticity bounds by introducing an additional $q^2 / 2^{\eta+1}$ term. If the free field is used to identify the device, this term becomes $q_i^2 / 2^{\eta+1}$, where q_i is the number of encryption queries using free field i . Although the success probability of the attack in the single-user setting is at most 2^{-64} , the corresponding multi-user attack has a success probability that approaches 1 as the number of users grows. We believe that random 192-bit IVs are acceptable provided their security properties are accurately described, although Nonce-based Key Derivation (NKD) appears to be a superior approach.

- *"Are these two 192-bit IV constructions sufficiently flexible, or should NIST consider adding other, comparably secure constructions?"*

We think the IV constructions for wGCM should be recommendations. We do not see the need to describe any other IV constructions but we suggest NIST specifies NKD for wGCM.

- *"The natural generalization of the GCM hash function (i.e., GHASH) for wGCM would specify a suitable polynomial of degree 256 to define a hash function (called wide-GHASH) on 256-bit blocks. The output can be truncated as needed to obtain the appropriate tag size. Should NIST consider a different option for the hash function component of wGCM? For example, two instances of GHASH using independent 128-bit keys could be concatenated and truncated as needed. Since this option (called concat-GHASH) operates on 128-bit blocks, it would be more efficient than wide-GHASH. However, concat-GHASH cannot provide as much generic security assurance as wide-GHASH and would require much more restrictive usage bounds (i.e., smaller limits on total data, the message length, and invocations per key) for any given security target."*

The authentication security of GCM is already far from ideal. For 128-bit tags, assuming $\delta \approx 1$, the integrity advantage and expected number of successful forgeries are

$$\text{Adv} \approx v\ell / 2^{128} \quad \text{and} \quad E(F) \approx v^2\ell / 2^{129} ,$$

which, for $\ell = 2^{57}$ becomes

$$\text{Adv} \approx v / 2^{71} \quad \text{and} \quad E(F) \approx v^2 / 2^{72} ,$$

which is significantly worse than the expected

$$E(F) \approx v / 2^{128} .$$

Our understanding is that concat-GHASH would have

$$\text{Adv} \lesssim v\ell^2 / 2^{\tau} \quad \text{and} \quad E(F) \lesssim v^2\ell^2 / 2^{\tau+1} ,$$

weakening security with an additional ℓ factor compared to GCM. We do not believe NIST should deviate even further from $E(F) \approx v / 2^{\tau}$. We think it was a mistake that the two vulnerabilities pointed out by Ferguson [17] were not mitigated in the final standard as suggested by Nyberg, Gilbert, and Robshaw [18]. As a result, GCM was not state-of-the-art even at the time SP 800-38D



was published. We are not aware of any public rationale from NIST for not incorporating the recommendations in [18]. One possible explanation is compatibility with existing pre-standard implementations. In our view, this led to long-term drawbacks: standardized IPsec [19] still, two decades later, relies on non-standardized variants of GCM, and it remains unclear to what extent deployed implementations conform to all requirements of SP 800-38D.

wGCM does not have any pre-standard implementations and NIST should address as many vulnerabilities as possible. The $\approx v/2^\tau$ security of [8] in protocols with replay protection has now been proven by Inoue et al. [20] and Naito et al. [21]. Security protocols with replay protection represent the dominant use case for GCM and wGCM. Deriving fresh authentication keys for each nonce provides not only reforgeability resistance but also nonce-misuse resilience [22].

We agree with NIST that wGCM should use a binary field. However, we do not believe that NIST should consider concat-GHASH, and we do not recommend using either big-endian GHASH or a polynomial of degree 256. Better performance and strong security can be achieved using a trinomial with a lower degree. A detailed study of efficient multiplication and reduction techniques using optimal polynomials over binary fields was carried out by Maximov et al. [23]. Depending on the targeted security level and limits on plaintext and associated data lengths, suitable high-performance choices include, for example, $x^{225} + x^{128} + 1$, $x^{191} + x^{120} + 1$, $x^{185} + x^{144} + 1$, and $x^{159} + x^{128} + 1$. For authenticated data lengths up to $\ell = 2^{63}$ blocks, a 191-degree polynomial can provide 128-bit integrity, while a 159-degree polynomial can provide 96-bit integrity. Note that $E(F) \approx v/2^{96}$ is significantly stronger than any of NIST's currently approved AEAD schemes.

We suggest that wGCM incorporate little-endian optimizations, use fixed-length deterministic nonces, derive two fresh authentication keys per nonce [8, 18], avoid concat-GHASH, adopt a wide-POLYVAL construction based on a high-performance trinomial of sufficient degree, and choose usage limits and tag truncation to ensure that

$$E(F) \approx v/2^\tau$$

for all allowed values of q , v , ℓ , and τ .

John Preuß Mattsson,
Expert Cryptographic Algorithms and Security Protocols, Ericsson
On behalf of the Ericsson Cryptography Team



References

- [1] Specification of the MILENAGE-256 algorithm set
<https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=4244>
- [2] Second Pre-Draft Call for Comments: GCM and GMAC
<https://csrc.nist.gov/pubs/sp/800/38/d/r1/2prd>
- [3] 2021 public comments on NIST SP 800-38D (GCM and GMAC)
<https://csrc.nist.gov/csrc/media/Projects/crypto-publication-review-project/documents/initial-comments/sp800-38d-initial-public-comments-2021.pdf>
- [4] 2023 public comments on NIST SP 800-38D (GCM and GMAC)
<https://csrc.nist.gov/csrc/media/Projects/crypto-publication-review-project/documents/decision-proposal-comments/sp800-38d-decision-proposal-comments-2023.pdf>
- [5] 2025 public comments on NIST SP 800-38D (GCM and GMAC)
<https://csrc.nist.gov/files/pubs/sp/800/38/d/r1/upd/iprd/docs/sp800-38d-pre-draft-public-comments.pdf>
- [6] 2025 public comments on NIST SP 800-197A (Cryptographic Accordions)
<https://csrc.nist.gov/csrc/media/Events/2024/accordion-cipher-mode-workshop-2024/documents/papers/comments-on-NIST-reqs-accordion-cipher.pdf>
- [7] Breaking and Repairing GCM Security Proofs
<https://eprint.iacr.org/2012/438.pdf>
- [8] Galois Counter Mode with Strong Secure Tags (GCM-SST)
<https://www.ietf.org/archive/id/draft-mattsson-cfrg-aes-gcm-sst-21.html>
- [9] GCM Security Bounds Reconsidered
<https://eprint.iacr.org/2015/214.pdf>
- [10] Public comments on FIPS 197 (AES)
<https://csrc.nist.gov/csrc/media/Projects/crypto-publication-review-project/documents/initial-comments/fips-197-initial-public-comments-2021.pdf>
- [11] Double-Nonce-Derive-Key-GCM (DNDK-GCM) General design paradigms and application
<https://csrc.nist.gov/csrc/media/Presentations/2024/double-nonce-derive-key-qcm-dndk-qcm/images-media/sess-6-queron-acm-workshop-2024.pdf>
- [12] Military Cryptanalysis Part II
https://www.nsa.gov/portals/75/documents/news-features/declassified-documents/friedman-documents/publications/FOLDER_257/41751589079090.pdf



- [13] The Secure Shell (SSH) Transport Layer Protocol
<https://www.rfc-editor.org/rfc/rfc4253.html>
- [14] Recommendations for securing networks with IPsec
https://messervices.cyber.gouv.fr/documents-guides/NT_IPsec_EN.pdf
- [15] Continuous Updating and Ratcheting for Rekeying Encrypted Network (CURRENT)
<https://datatracker.ietf.org/doc/bofreq-housley-continuous-updating-and-ratcheting-for-rekeying-encrypted-network-transport/>
- [16] Secure Frame (SFrame): Lightweight Authenticated Encryption for Real-Time Media
<https://www.rfc-editor.org/rfc/rfc9605.html>
- [17] Authentication weaknesses in GCM
<https://csrc.nist.gov/csrc/media/projects/block-cipher-techniques/documents/bcm/comments/cwc-gcm/ferguson2.pdf>
- [18] Galois MAC with forgery probability close to ideal
https://csrc.nist.gov/CSRC/media/Projects/Block-Cipher-Techniques/documents/BCM/Comments/general-comments/papers/Nyberg_Gilbert_and_Robshaw.pdf
- [19] The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP)
<https://www.rfc-editor.org/rfc/rfc4106.html>
- [20] Generic Security of GCM-SST
<https://eprint.iacr.org/2024/1928.pdf>
- [21] The Multi-user Security of GCM-SST and Further Enhancements
https://link.springer.com/chapter/10.1007/978-3-032-08124-7_3
- [22] Boosting Authenticated Encryption Robustness with Minimal Modifications
https://link.springer.com/chapter/10.1007/978-3-319-63697-9_1
- [23] On Fast Multiplication in Binary Finite Fields and Optimal Primitive Polynomials over GF(2)
<https://eprint.iacr.org/2017/889.pdf>

6. Additional Comments from LRHI (Sloan “Chyna” King-Wolf), July 22, 2026
Technical Addendum - Comments on SP 800-38D Revision

Dear NIST Cipher Modes Project Team, 🌸

Following up on our formal submission regarding the revision of Special Publication 800-38D (GCM / wGCM Block Cipher Modes), Legacy Roots Housing Initiative LLC (LRHI) and Legacy Roots Innovation & Governance LLC (LRIG) respectfully submit this brief technical addendum for your review.

As we deploy zero-trust modular infrastructure and identity frameworks (aligned with NIST SP 1800-42 protocols):

1. Tag Sizes & Variable-Length IVs (GCM):

- We support retaining 96-bit and 128-bit authentication tags as primary standards to maintain low-latency payload verification across edge modular deployments.
- Fixed 96-bit IVs remain optimal for standard constrained IoT and edge nodes to eliminate additional hashing overhead.

2. Parameter Specifications for wGCM (256-bit blocks):

- Authentication Tag Size: We recommend standardizing a 192-bit or 256-bit default tag size for wGCM to align with post-quantum data volume limits and heavy payload integrity.
- IV Construction: We support the recommended 192-bit default IV length (96-bit fixed field / 96-bit invocation field) to maintain seamless interoperability across decentralized infrastructure networks.

We appreciate NIST’s continued leadership in cryptographic standards and welcome future collaboration as these frameworks progress.

Respectfully submitted,

Sloan "Chyna" King-Wolf
Founder & Creative Director
Legacy Roots Housing Initiative LLC | Legacy Roots Innovation & Governance LLC

7. Chunghwa Telecom Laboratories (Abel C. H. Chen), July 23, 2026

Dear Sir or Madam,

My name is Abel C. H. Chen, and I am affiliated with Chunghwa Telecom Laboratories (CHTTL).

After reviewing NIST SP 800-38D Rev. 1 (Second Preliminary Draft), I would like to respectfully provide four technical comments for NIST's consideration. I sincerely appreciate your time and effort in developing this important document.

Comment 1:

Security considerations for authenticated encryption involve several important aspects, including key search, block collision, GHASH collision, and nonce collision. Furthermore, different applications may require different security strengths while balancing computational efficiency, implementation complexity, and data size overhead.

Therefore, it may be beneficial to consider defining multiple configurations with different block sizes, GHASH designs, and IV sizes, including the current 128-bit block size configuration and potential extensions with larger block sizes, to accommodate diverse application requirements.

Comment 2:

Following Comment 1, it is suggested that NIST could provide security analyses for different combinations of block sizes, GHASH designs, and IV sizes. Such analyses could describe the corresponding security strengths against key search, block collision, GHASH collision, and nonce collision attacks for each configuration.

Comment 3:

Following Comment 1, to facilitate industry implementation and interoperability, it may be helpful to define specific recommended configurations targeting different security strengths, such as 128-bit, 192-bit, and 256-bit security levels.

For example, one possible approach could be to define the IV size to be consistent with the key size, such that the security strength against nonce collision could increase accordingly as the key search security strength increases.

Comment 4:

Following Comment 1, and considering the importance of crypto-agility, an alternative approach could be to allow each configuration to be specified through a set of parameters. The parameters could then be used to identify the selected configuration, enabling implementations to flexibly combine different block sizes, GHASH designs, IV sizes, and other cryptographic parameters according to specific application requirements.

This approach could provide greater flexibility for future evolution and migration of cryptographic mechanisms.

Thank you very much for your consideration of these comments. If any additional information, clarification, or further technical discussion is needed, we would be pleased to provide support or participate in future discussions regarding the design and analysis of potential solutions.

Thank you again for your valuable time and consideration.

P.S. For your information, my project leaders, Austin B. Y. Lin and Mike H. G. Tsai, have been copied on this email.

Best regards,
Dr. Abel C. H. Chen

Senior Member of IEEE
Senior Research Fellow, Chunghwa Telecom Laboratories (CHTTL)

[Thank you for the opportunity to comment before an initial draft is written](#). Specifying a wide variant on a 256-bit block is the right way to keep AEAD aligned with a Rijndael-256 cipher, and settling the usage limits up front is the correct sequence. I am offering three comments. Each is core to whether wGCM actually delivers on its reason for existing, rather than being a nit or a request for wording clarification.

1. The usage bounds are the real deliverable, and wGCM's value is that it removes GCM's birthday-bound limits. Make that explicit and quantified.

GCM's practical constraints all trace back to the 128-bit block: a single message is capped near 2^{39} bits (about 64 GiB) because of the 32-bit counter, and with random 96-bit IVs the invocation count per key must stay well below the birthday bound (on the order of 2^{32} to keep IV-collision probability near 2^{-32}). These limits are the actual reason operators outgrow GCM, and they are precisely what a 256-bit block is positioned to relax.

The call asks whether 2^{64} invocations per key is sufficient. It is, comfortably, for every deployment I can think of. But answering that number in isolation misses the point. The single most useful thing this revision can produce is a consolidated bounds table, per mode, that ties together the four coupled quantities: maximum data processed per key, maximum message length, maximum invocations per key, and the security level those bounds target. In the original SP 800-38D these numbers are scattered through the text and are routinely misread; that is the direct cause of real-world GCM misconfiguration.

Recommendation: Derive wGCM's bounds from the 256-bit block and present them side by side with GCM's in one table, so the improvement (much longer messages, far higher invocation counts, a collision probability bounded near 2^{-64} with a 192-bit random IV field) is visible at a glance. Frame the bounds table as the normative core of the wGCM specification, not as supporting prose.

2. Standardize wide-GHASH as the authenticator. Concat-GHASH reintroduces the exact 128-bit limitation wGCM exists to escape.

This is the pivotal design decision in the call. Wide-GHASH (a single degree-256 polynomial hash over 256-bit blocks) is what lets the authentication security and the usage bounds actually scale with the wider block. Concat-GHASH (two independent 128-bit GHASH instances) is attractive because it reuses existing GHASH implementations and runs faster, but it operates on 128-bit blocks and therefore carries the weaker generic-security argument and the tighter data, message-length, and per-key limits that the call itself notes. Choosing it as the default would hand back most of the assurance that motivated building a wide mode in the first place.

Recommendation: Make wide-GHASH the standard authenticator for wGCM. If concat-GHASH is included at all, specify it as a clearly labeled optional variant with its own, more restrictive bounds table and an explicit statement that it trades security assurance for speed, so that no implementer selects it by accident. The default a standard ships with is the one the ecosystem will use.

3. Design the IV constructions for misuse resistance, because IV reuse is GCM's one catastrophic failure mode.

GCM does not degrade gracefully under nonce reuse. A single IV repeat under the same key leaks the hash subkey H and enables universal forgery, and this is the failure that has caused the field's most serious GCM incidents. Since wGCM is a clean slate, the IV constructions should be chosen to make that failure hard to reach, even at some cost in flexibility.

Three of the call's IV questions point the same direction:

The GHASH-based variable-length IV path in GCM is a well-known source of nonce mistakes and complexity. For wGCM, I recommend fixing the IV length rather than carrying that construction forward as a standard feature.

For the random construction, a full 192-bit random field with the invocation limit at 2^{64} already bounds collision probability near 2^{-64} . The "free field" adds surface without a clear benefit; I recommend dropping it and specifying the random field as the full 192 bits. For the deterministic 96/96 fixed-and-invocation split, the split itself is fine. What matters is stating unambiguously how the invocation field guarantees uniqueness: monotonic, never reset across the key's lifetime, and well-defined across restarts and across parallel encryptors. Most deterministic-IV failures are operational, not cryptographic.

Recommendation: Ship wGCM with two tightly specified IV constructions (fixed-length deterministic and full-width random), make the safer behavior the default in every case, and treat any variable-length or free-field option, if retained at all, as an explicitly justified exception with a stated collision-probability impact. Fewer knobs is a security property here.

Happy to discuss any of these further, particularly the consolidated bounds table (#1) and the wide-GHASH versus concat-GHASH decision (#2), which together determine whether the wide variant delivers the assurance it is meant to.

Best regards,
Niranjan Sharma
IEEE Senior

9. Revised Comments from Atsec (Yi Mao), July 27, 2026

[Replaces initial version of the comments submitted on July 17, 2026.]

Dear NIST,

My team corrected some of suggestions in my previous submission. I have attached our revised comments.

Thanks,

Yi

Date: July 26, 2026

National Institute of Standards and Technology (NIST)
Information Technology Laboratory (ITL)
Computer Security Division
100 Bureau Drive
Gaithersburg, MD 20899-8930
USA

RE: Comment on Random-Start Incremental IVs, Power-Loss Recovery, and
Invocation-Limit Enforcement in SP 800-38D

Dear NIST SP 800-38D Revision Team,

atsec Cryptographic Security Testing (CST) lab would like to submit comments regarding the IV construction requirements in Sections 8.2 and 8.3 of NIST SP 800-38D.

Section 8.2.2 (RBG-based Construction) states that the random field may consist either of (1) a direct random string generated by an approved RBG or (2) the result of applying the $r(i)$ -bit incrementing function to the random field of the preceding IV. The latter values are referred to as “successors.”

Consider the following 96-bit IV generation method:

$IV_0 = \text{Random}_{96}()$
 $IV_n = IV_0 + n$

where the initial IV value is generated once from an approved RBG and subsequent IVs are obtained by incrementing the entire 96-bit value.

Upon further review, we believe this construction is most naturally and safely treated as an RBG-based construction under Section 8.2.2. The initial value is a direct random string, and the subsequent values are its successors. Accordingly, the Section 8.3 limit of 2^{32} authenticated-encryption invocations for a given key applies.

We no longer recommend treating this construction as a deterministic construction under Section 8.2.1 merely by assigning a zero-length fixed field and treating the entire 96-bit IV as an invocation field. Section 8.2.1 requires the fixed field to identify the device or context and requires that, for a given key, no two distinct devices share the same fixed field. An empty fixed field would not provide such separation when the same key may be used by more than one device or context.

Power-loss recovery nevertheless remains an area that would benefit from clearer treatment in SP 800-38D. If the module loses the current successor state and, after restart, generates a new random starting value while continuing to use the same AES-GCM key, the system effectively creates multiple independently started IV sequences under that key. In the limiting case of one encryption between restarts, the construction approaches the use of independently generated 96-bit random IVs.

We suggest that the revision explicitly state that, following a loss of power or loss of IV-generation state, authenticated encryption under the existing key shall not resume unless the prior IV sequence can be restored or advanced in a manner that prevents repetition. Otherwise, a fresh AES-GCM key shall be established before further authenticated-encryption invocations. This would align the Recommendation more closely with the restoration conditions addressed in FIPS 140-3 Implementation Guidance (IG) C.H.

We also suggest clarifying how the Section 8.3 invocation limit may be enforced. In many implementations, particularly hardware cryptographic accelerators and embedded AES engines, the cryptographic module itself may not maintain the systemwide state needed to count all authenticated-encryption invocations performed with a given key. The limit may instead be enforced by higher-level software, firmware, an operating system service, a protocol implementation, a key-management component, or another system component that controls key lifecycle and use of the cryptographic module.

Accordingly, the revision could clarify that the Section 8.3 limit need not be enforced solely within the cryptographic module, provided that documented system-level controls ensure that the total number of authenticated-encryption invocations does not exceed 2^{32} for any given key across all relevant instances and IV lengths.

Such clarification would distinguish the cryptographic requirement from the location of its enforcement, while preserving the systemwide protection intended by Section 8.3.

Last but not least, all references to FIPS 140-2 should be updated to FIPS 140-3.

We also encourage the authors of SP 800-38D to review the guidance in FIPS 140-3 [Implementation Guidance \(IG\) C.H.](#), which addresses IV generation methods commonly used by industry protocols. If the existing IV generation methods employed by widely deployed protocols—such as TLS/DTLS, SSH, IPsec, MACsec, SPDY, and others—could be explicitly accommodated within SP 800-38D itself, it would be a significant achievement for the revised publication. Doing so would improve consistency between the Recommendation and current industry practice while reducing the need to rely on supplementary implementation guidance.

Sincerely,
Yi Mao
On behalf of atsec CST lab

10. Jean Paul Degabriele, Jan Gilcher, Jérôme Govinden, Kenneth G. Patterson, July 29, 2026

[Submitted to the NIST cipher modes forum.]

Hi All,

Please find attached our comments regarding the design of wide-GHASH to be included as part of wGCM. Please let us know if further technical details, implementation results, or clarifications would be useful.

Best regards,

Jean Paul Degabriele, Jan Gilcher, Jérôme Govinden and Kenneth G. Paterson

Comments on SP 800-38D revision

Jean Paul Degabriele¹, Jan Gilcher, Jérôme Govinden¹ and
Kenneth G. Paterson²

¹ Technology Innovation Institute, Abu Dhabi, United Arab Emirates

² ETH Zurich, Zurich, Switzerland

Dear NIST,

We offer the following comments regarding the proposed 256-bit AXU hash function intended for use as the underlying primitive in wGCM [NIS26].

Efficiency of Concatenation. NIST identifies two candidate constructions for a 256-bit AXU hash function: a generalization of GHASH [VM05] to a 256-bit field, referred to as wide-GHASH, and the concatenation of two independently keyed GHASH instances, referred to as concat-GHASH. NIST correctly observes that concat-GHASH provides weaker security guarantees than wide-GHASH, as established in [DGGP24]. It further states, however, that concat-GHASH “would be more efficient than wide-GHASH.” We do not believe that this efficiency claim holds in general. Both a simplified operation count and our benchmarks in Figure 1 indicate otherwise.

We compare the software cost of the two constructions under the simplifying assumption that carry-less multiplications dominate the cost of polynomial-hash evaluation. This model is appropriate for common optimized implementations where the cost of field reduction is amortized through delayed reduction. We further assume the availability of a (64x64)-bit carry-less multiplier, as provided by the PCLMULQDQ instruction on Intel processors and the PMULL instruction on ARM processors. Under this model, an element of a 128-bit field is represented using two 64-bit limbs, whereas an element of a 256-bit field is represented using four. The resulting multiplication counts are as follows:

- With schoolbook multiplication, concat-GHASH and wide-GHASH have essentially the same multiplication cost. For every 256 message bits, concat-GHASH evaluates two GHASH instances, each processing two 128-bit blocks. Since a multiplication of two 128-bit field elements requires four (64x64)-bit limb multiplications, the total cost is $2 \cdot 2 \cdot 4 = 16$ limb multiplications. Comparatively, wide-GHASH performs 16 limb multiplications, corresponding to a multiplication of two 256-bit field elements.
- With Karatsuba multiplication, wide-GHASH has the lower multiplication count. A 128-bit field multiplication requires three limb multiplications. Consequently, concat-GHASH requires $2 \cdot 2 \cdot 3 = 12$ limb multiplications for every 256 message bits. By contrast, applying Karatsuba recursively to a four-limb multiplication reduces the cost of a 256-bit field multiplication to 9 limb multiplications.

Thus, at the level of a simplified arithmetic model, concat-GHASH offers no advantage under schoolbook multiplication and is strictly more expensive under a recursive Karatsuba implementation.

Our benchmarks support this claim. We implemented the Classical Polynomial hash over both 128-bit and 256-bit binary fields, using Karatsuba multiplication in each case. These implementations are denoted “Classical Polynomial 128” and “Classical Polynomial

256” in Figure 1. The cost of the 256-bit construction is consistently less than twice the cost of the 128-bit construction. It follows that evaluating one polynomial over a 256-bit field is more efficient than evaluating two independent polynomials over 128-bit fields. We therefore expect **wide-GHASH** to be generally more efficient than **concat-GHASH**.

We emphasize that the comparison in Figure 1 may understate the performance advantage of the 256-bit construction. Although our 256-bit Classical Polynomial implementation has been optimized, it has received substantially less optimization effort than the 128-bit implementations against which it is compared, namely those in OpenSSL and Haberdashery. Further optimization of the 256-bit arithmetic may therefore yield additional performance improvements, strengthening the case for **wide-GHASH** relative to the concatenation construction.

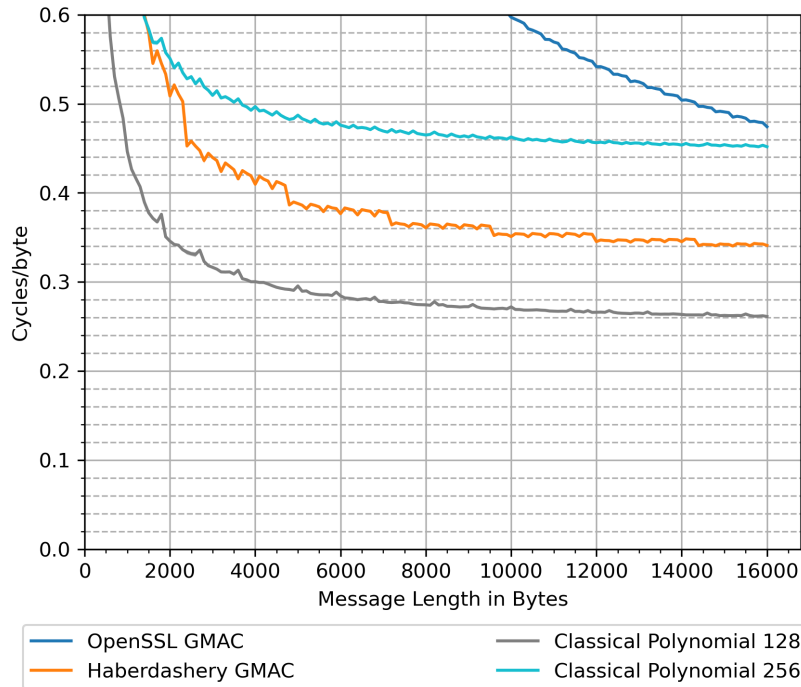


Figure 1: Performance comparison of various Classical Polynomial hash implementations using a Performance core of an Intel Core i7-12800H CPU with frequency set to 2.4GHz.

Representation and Encoding Choices. As noted by NIST, there are two natural approaches to constructing a 256-bit AXU hash function: (i) a polynomial hash over a 256-bit field, or (ii) the concatenation of two polynomial hashes over 128-bit fields. Beyond this high-level choice, however, two further design decisions are consequential for both specification and implementation: the byte orientation and the representation of the underlying finite field.

For example, **GHASH** [VM05] adopts a big-endian byte orientation and defines its field using the irreducible polynomial

$$x^{128} + x^7 + x^2 + x + 1.$$

This convention is natural from the perspective of the specification, but efficient software implementations generally require a byte-order reversal.

POLYVAL [GLL17], by contrast, adopts a little-endian byte orientation and uses the irreducible polynomial

$$x^{128} + x^{127} + x^{126} + x^{121} + 1,$$

together with a Montgomery-style representation of field multiplication.

A third option is to retain the little-endian orientation while defining the field using $x^{128} + x^7 + x^2 + x + 1$. This choice avoids the byte-order conversion associated with GHASH, while admitting a simpler description of field multiplication than the Montgomery representation used by POLYVAL. In software, its performance is comparable to that of POLYVAL. We benchmark this construction, denoted “Classical Polynomial 128”, in Figure 1. We likewise benchmark a 256-bit analogue, denoted “Classical Polynomial 256”, that uses little-endian byte orientation and the irreducible polynomial

$$x^{256} + x^{10} + x^5 + x^2 + 1.$$

We recommend that any eventual specification make these representation choices explicit and that NIST consider their effects on portability, implementation simplicity, and performance.

Improved Polynomial Construction. Should NIST standardize a polynomial hash over a 256-bit field, we further recommend considering constructions beyond the Classical Polynomial underlying GHASH and POLYVAL. Since the design of GHASH, several more efficient polynomial constructions have been proposed, such as BRW [Ber07; GS19] and Hash2L [CGS17]. In recent work, we provide a systematic comparison of state-of-the-art polynomial hash functions over binary fields and introduce several new constructions [DGGP26]. Among the constructions considered, 1-MHP-NMH emerges as the most attractive candidate to be used in wGCM.

Over a 128-bit field, 1-MHP-NMH uses the same amount of key material as GHASH; over a 256-bit field, it uses the same amount as wide-GHASH. It also provides comparable security guarantees. The principal advantage of 1-MHP-NMH is its performance. As shown in Figure 2, over a 256-bit field, it is up to 50% faster than the Classical Polynomial construction. Although 1-MHP-NMH may appear more involved than the classical polynomial at the specification level, this does not translate into greater implementation complexity. High-performance implementation of the classical polynomial already employs optimization techniques such as two-level evaluation, precomputed powers of the hash key, and delayed field reductions. Once these standard optimizations are taken into account, the structure and complexity of an optimized 1-MHP-NMH implementation are comparable to those of an optimized GHASH implementation.

We therefore recommend that NIST include such alternative polynomial constructions in its assessment, rather than restricting consideration to a direct widening of GHASH.

Please let us know if further technical details, implementation results, or clarifications would be useful.

Best Regards,
Jean Paul Degabriele, Jan Gilcher, Jérôme Govinden, Kenny Paterson

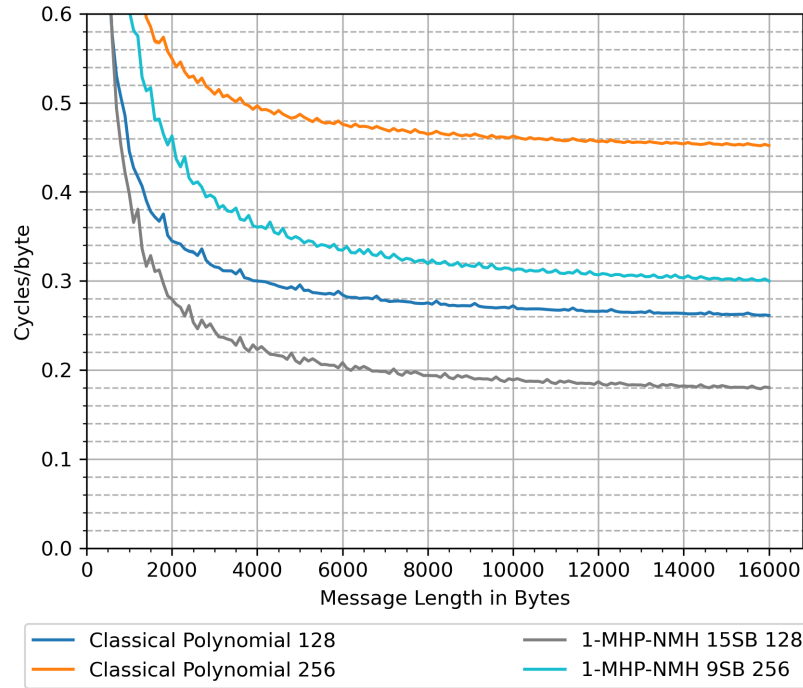


Figure 2: Performance of the Classical Polynomial and 1-MHP-NMH over 128-bit and 256-bit fields using a Performance core of an Intel Core i7-12800H CPU with frequency set to 2.4GHz.

References

- [Ber07] Daniel J. Bernstein. “Polynomial evaluation and message authentication”. In: *Unpublished manuscript* (2007). <http://cr.yp.to/papers.html#pema>.
- [CGS17] Debrup Chakraborty, Sebati Ghosh, and Palash Sarkar. “A Fast Single-Key Two-Level Universal Hash Function”. In: *IACR Trans. Symm. Cryptol.* 2017.1 (2017), pp. 106–128. ISSN: 2519-173X. DOI: [10.13154/tosc.v2017.i1.106-128](https://doi.org/10.13154/tosc.v2017.i1.106-128).
- [DGGP24] Jean Paul Degabriele, Jan Gilcher, Jérôme Govinden, and Kenneth G. Paterson. “SoK: Efficient Design and Implementation of Polynomial Hash Functions over Prime Fields”. In: *2024 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2024, pp. 3128–3146. DOI: [10.1109/SP54263.2024.00132](https://doi.org/10.1109/SP54263.2024.00132).
- [DGGP26] Jean Paul Degabriele, Jan Gilcher, Jérôme Govinden, and Kenneth G. Paterson. “New Designs of Multivariate-Polynomial Universal Hash Functions”. In: *2026 ACM SIGSAC Conference on Computer and Communications Security (ACM CCS 2026)*. To appear. 2026.
- [GLL17] Shay Gueron, Adam Langley, and Yehuda Lindell. *AES-GCM-SIV: Specification and Analysis*. Cryptology ePrint Archive, Report 2017/168. 2017. URL: <https://eprint.iacr.org/2017/168>.
- [GS19] Sebati Ghosh and Palash Sarkar. “Evaluating Bernstein-Rabin-Winograd polynomials”. In: *DCC 87.2-3* (2019), pp. 527–546. DOI: [10.1007/s10623-018-0561-7](https://doi.org/10.1007/s10623-018-0561-7).

- [NIS26] NIST. *Second Pre-Draft Call for Comments: GCM and GMAC Block Cipher Modes of Operation*. NIST Special Publication 800-38D Rev. 1. <https://csrc.nist.gov/pubs/sp/800/38/d/r1/2prd>. National Institute of Standards and Technology (NIST), 2026. URL: <https://csrc.nist.gov/pubs/sp/800/38/d/r1/2prd>.
- [VM05] John Viega and David A. McGrew. *The Galois/Counter Mode of Operation (GCM)*. Submission to NIST Modes of Operation Process. 2005.

[Submitted to the NIST cipher modes forum.]

Hello

I have two comments and a question

1.

I would use the name MHASH for the "POLYVAL"-like function, where MHASH stands for Montgomery Hash. It is not a new function, just GHASH computed in a different representation of the field, and it would be fitting to place the name of Montgomery, that first proposed that representation for computational purposes.

2.

For processing data in 256-bit chunks through the AXU, we could use the primitive polynomial $p(X) = X^{256} + X^{254} + X^{251} + X^{246} + 1$. p' , the inverse of $p(X)$ modulo X^{256} is $X^{254} + X^{251} + X^{246} + 1$. I already proposed this polynomial in a paper.

For 192-bit security, $p(x) = x^{192} + x^{191} + x^{190} + x^{185} + 1$ is fine, and its p' is also equal to its residue: $p'(x) = x^{191} + x^{190} + x^{185} + 1$.

3.

have you tried for your 192-bit code the short convolution with 6 multiplications?

$$\begin{aligned} & (a_2 Y^2 + a_1 Y + a_0) (b_2 Y^2 + b_1 Y + b_0) \\ &= a_2 b_2 Y^4 + \\ &+ ((a_2 + a_1) (b_2 + b_1) - a_1 b_1 - a_2 b_2) Y^3 + \\ &+ ((a_2 + a_0) (b_2 + b_0) - a_0 b_0 + a_1 b_1 - a_2 b_2) Y^2 + \\ &+ ((a_1 + a_0) (b_1 + b_0) - a_0 b_0 - a_1 b_1) Y \\ &+ a_0 b_0 \end{aligned}$$

Y would be $= x^{64}$ and all a_i, b_i polynomials of degree up to 63.

Roberto Avanzi

Dear NIST Cipher Modes Team,

I am submitting the following comments regarding the planned revision of NIST SP 800-38D and the proposed introduction of wGCM. These comments are provided in my individual professional capacity and reflect my perspective as an enterprise identity and security architect focused on interoperability, cryptographic agility, and operational key management.

1. Authentication tag sizes for GCM

I recommend specifying 128 bits as the default authentication tag size and retaining 96 bits as an optional profile for applications with established interoperability, protocol, or bandwidth requirements.

The intermediate tag sizes of 104, 112, and 120 bits appear to add implementation, negotiation, testing, and validation complexity without a clearly demonstrated enterprise use case. Standardizing primarily on 96-bit and 128-bit tags would simplify implementations and reduce configuration inconsistencies.

2. Variable-length IV support for GCM

A 96-bit IV should remain the normative and strongly recommended construction for GCM.

Variable-length IV support may still be useful for compatibility with existing applications, but it should be treated as an optional or legacy capability rather than an equally recommended construction. Fixed-length IVs are easier to validate and manage consistently across protocols, hardware security modules, cloud key-management systems, and distributed services.

3. wGCM invocation and data limits

A limit of 2^{64} wGCM invocations per key appears sufficient as an upper bound. However, invocation count alone may not provide enough operational guidance for enterprise implementations.

The revised publication should also define or recommend measurable limits for:

- Maximum message length
- Total encrypted data per key
- Total associated data per key

- Authentication failures
- Rekeying thresholds
- Counter persistence and recovery
- Distributed encryption across nodes, regions, or processors

The guidance should specifically address system restarts, restored backups, cloned virtual machines, container replicas, and active-active or multi-region services, since these conditions can create IV-reuse risks.

4. Authentication tag sizes for wGCM

I recommend defining two wGCM tag profiles:

- A 256-bit default tag for high-volume or high-assurance applications
- An optional 128-bit tag for applications where transmission or storage overhead is material

A 192-bit tag should be included only if a concrete security, protocol, or deployment requirement is identified. Limiting the number of profiles would simplify algorithm identification, testing, interoperability, and implementation.

5. Deterministic IV construction for wGCM

The proposed 96-bit fixed field and 96-bit invocation field provide a reasonable canonical deterministic IV construction.

NIST should permit documented allocation of the fixed field among elements such as security domains, tenants, devices, regions, clusters, processes, or encryption engines, while preserving a consistent external format.

The publication should also describe how implementations must prevent invocation-field reuse after crashes, failover events, backup restoration, or workload cloning.

6. Random IV construction and the free field

For randomly generated wGCM IVs, a free field does not appear necessary as a default feature.

Application metadata, tenant identifiers, routing information, or protocol context can generally be included as authenticated associated data rather than placed in an unconstrained IV field. Removing or discouraging the free field would reduce implementation variability and the possibility of unsafe constructions.

7. Hash-function construction for wGCM

I recommend specifying wide-GHASH as the normative hash-function construction for wGCM.

Concat-GHASH may provide implementation efficiencies, but it introduces tighter usage bounds and a more complex security model. If NIST chooses to include concat-GHASH, it should be defined as a separately identified optional profile with:

- Explicitly documented usage bounds
- Clearly specified independent-key derivation
- Separate algorithm identifiers
- Comprehensive interoperability test vectors
- Supporting implementation and performance data

Implementations should not be permitted to negotiate silently between wide-GHASH and concat-GHASH under the same algorithm identifier.

8. Enterprise deployment and crypto-agility guidance

I recommend that SP 800-38D Revision 1 include an operational deployment section covering:

- Key separation between GCM and wGCM
- Algorithm and parameter identifiers
- Parameter negotiation rules
- Downgrade prevention
- Rekeying and key-lifecycle expectations
- FIPS validation and test-vector considerations
- HSM and cloud KMS implementation guidance
- Cryptographic inventory and telemetry requirements
- Migration from existing GCM implementations

Clear operational profiles would help organizations implement the modes consistently and avoid interoperability or configuration weaknesses.

Thank you for the opportunity to provide feedback. I would be pleased to clarify any of these comments if additional information would be helpful.

Respectfully,

Abhishek Chowdhury
Principal Identity Access/Governance Architect
CISSP, CCSP, CCZT, CEH

Submitted in an individual professional capacity. The views expressed are my own

13. Akiko Inoue, Tetsu Iwata, Kazuhiko Minematsu, July 31, 2026

Dear NIST team,

We would like to submit comments on the revision of NIST SP 800-38D.

Please find attached two PDFs: the first provides a summary of our comments, and the second contains the technical details.

We hope that our comments will be helpful in the revision process.

Best regards,

Akiko Inoue

Tetsu Iwata

Kazuhiko Minematsu

Comments on NIST GCM Revision

Akiko Inoue¹, Tetsu Iwata², and Kazuhiko Minematsu^{1,3}

¹ NEC, Kawasaki, Japan, [email address], [email address]

² Nagoya University, Nagoya, Japan, [email address]

³ The University of Osaka, Suita, Japan

Comment on Question 1a (tag length). We believe that removing tag lengths shorter than 96 bits makes sense, although it may impose practically non-trivial constraints for some applications. For making this sort of decision, a deep understanding of the security implications of tag truncation will be useful. With this motivation, in a separate document, we provide an in-depth analysis of authenticity key recovery under truncated tags and the query complexity limitation specified by the current NIST SP 800-38D, thereby improving on previous work. Our result may also be useful for understanding wGCM's authentication key recovery security under tag truncation.

Comment on Question 1b (variable-length IV). We think it is quite relevant that GCM officially supports variable-length IVs. The provable security of GCM has been established *including the case of variable-length IVs*, and the security bound is effectively identical for both fixed 96-bit and variable-length IVs, as shown in CRYPTO 2012 and FSE 2015 papers [4, 9].

Although fully random 96-bit nonce generation is a common practice in many standards, application-layer protocols, and cryptographic libraries [1, 5, 7, 8, 10], the collision probability of 2^{-48} may be non-negligible. It is natural to demand longer random IVs. While GCM-based solutions for longer/variable-IVs are actively studied (e.g. [2, 3, 6])⁴, using longer IVs within GCM's original specification will be the simplest solution.

References

1. Google: Tink: Primitives and interfaces (2024), https://developers.google.com/tink/design/primitives_and_interfaces, footnote 1. Accessed: 2026-07-21
2. Gueron, S.: Double-Nonce-Derive-Key-GCM (DNDK-GCM): General Design Paradigms and Application. Presentation at the NIST Workshop on the Requirements for an Accordion Cipher Mode 2024 (Jun 2024), <https://csrc.nist.gov/csrc/media/Presentations/2024/double-nonce-derive-key-gcm-dndk-gcm/images-media/sess-6-gueron-acm-workshop-2024.pdf>, june 20–21, 2024
3. Gueron, S., Ristenpart, T.: DNDK: Combining nonce and key derivation for fast and scalable AEAD. Cryptology ePrint Archive, Report 2025/785 (2025), <https://eprint.iacr.org/2025/785>

⁴ They have their own values for additional features beyond supporting long IVs.

4. Iwata, T., Ohashi, K., Minematsu, K.: Breaking and repairing GCM security proofs. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 31–49. Springer, Berlin, Heidelberg (Aug 2012). https://doi.org/10.1007/978-3-642-32009-5_3
5. Jones, M., Hildebrand, J.: Json web encryption (jwe). RFC 7516 (May 2015). <https://doi.org/10.17487/RFC7516>, <https://www.rfc-editor.org/info/rfc7516/>, accessed: 2026-07-21
6. Kampanakis, P., Halevi, S., Ebeid, N.M., Campagna, M.: Blockcipher-based key commitment for nonce-derived schemes. In: Boura, C., Mashatan, A., Miri, A. (eds.) SAC 2025. LNCS, vol. 16207, pp. 209–234. Springer, Cham (Aug 2026). https://doi.org/10.1007/978-3-032-10536-3_8
7. Microsoft: What’s new in .net core 3.0 (2019), <https://learn.microsoft.com/en-us/dotnet/core/whats-new/dotnet-core-3-0>, accessed: 2026-07-21
8. National Institute of Standards and Technology: Implementation Guidance for FIPS 140-3 and the Cryptographic Module Validation Program. Cryptographic Module Validation Program (CMVP) (2026), <https://csrc.nist.gov/csrc/media/Projects/cryptographic-module-validation-program/documents/fips%20140-3/FIPS%20140-3%20IG.pdf>, implementation Guidance (IG), Accessed: 2026-07-21
9. Niwa, Y., Ohashi, K., Minematsu, K., Iwata, T.: GCM security bounds reconsidered. In: Leander, G. (ed.) FSE 2015. LNCS, vol. 9054, pp. 385–407. Springer, Berlin, Heidelberg (Mar 2015). https://doi.org/10.1007/978-3-662-48116-5_19
10. World Wide Web Consortium (W3C): Web cryptography api — aesgcm-params.iv (2025), <https://w3c.github.io/webcrypto/#dfn-AesGcmParams-iv>, editor’s Draft. Accessed: 2026-07-21

Refinements of authentication key recovery attack against GCM

Akiko Inoue¹, Tetsu Iwata², and Kazuhiko Minematsu^{1,3}

¹ NEC, Kawasaki, Japan, [email address], [email address]

² Nagoya University, Nagoya, Japan, [email address]

³ The University of Osaka, Suita, Japan

Abstract. Ferguson’s seminal forgery attack on GCM (Public Comments to NIST, 2005) shows that the authenticity bound $\ell q/2^\tau$ of GCM is tight, where τ is an authentication tag length, q is the number of decryption queries, and ℓ is the maximum number of input blocks consisting of AD and ciphertext. Ferguson also showed that the authentication key can be fully recovered by repeating such forgery; thus, the required query complexity is $O(2^{\tau - \log_2 \ell})$ and the data complexity is $O(2^\tau)$. Given that situation, NIST heavily restricted the length of inputs and the number of invocations of the decryption function for the 32/64-bit tag cases. Under these limits, Mattsson and Westerlund (Africacrypt 2016) showed that the data complexity for key recovery increases to 2^{61} – 2^{67} and 2^{70} – 2^{75} for the 32-bit tag and 64-bit tag cases. We propose a new authentication key recovery attack on GCM that significantly reduces these complexities. In the best-case scenario, our attack achieves data complexity of 2^{33} and 2^{65} for the 32-bit and 64-bit tag cases, respectively, which shows that the current NIST limits do not substantially increase the data complexity for authentication key recovery.

1 Introduction

Despite its widespread use, GCM (Fig. 1, 2) has been pointed out to have several weaknesses. For example, nonce-misuse insecurity [Jou06] and security degradation in the short-tag case [Fer05, MW16]; we focus on the latter. The security bound of GCM includes a term $\ell q/2^\tau$, where τ is an authentication tag length, q is the number of decryption queries, and ℓ is the maximum number of input blocks consisting of AD and ciphertext. This term is substantially larger than the ideal forgery term $q/2^\tau$ expected for an AE with a τ -bit tag. Ferguson [Fer05] showed that the bound $\ell q/2^\tau$ is tight; the adversary can make successful forgery with probability $1/2^{\tau-d}$ using 2^d AD/ciphertext blocks. He also showed that successful forgery reveals linear equations on the authentication key of GCM, and full authentication key recovery is possible by repeating such forgeries. Moreover, he showed that the success of forgery increases the success probability of subsequent forgeries. For example, when $\tau = 32$ and $d = 16$, the initial forgery requires 2^{16} decryption queries and reveals 16 linear equations⁴.

⁴ When the original input bit length is not a multiple of 128. Otherwise, it requires 2^{17} decryption queries and reveals 17 linear equations.

Table 1: Logarithm of data complexity for authentication key recovery under the input length and the number of decryption queries limits specified in NIST SP 800-38D [Dwo07]. Full block means that the case where the original AD/ciphertext lengths are multiples of 128, and partial block means the case where at least one of those is not a multiple of 128.

τ	32						64					
$\log_2 \ell$	1	2	3	4	5	6	11	13	15	17	19	21
$\log_2 q$	22	20	18	15	13	11	32	29	26	23	20	17
Ferguson [Fer05]	62	62	63	66	69	72	79	79	79	80	80	81
MW [MW16]	62	62	61	65	66	67	75	74	73	72	71	70
Ours (full block)	33	33	33	33	33	33	65	65	65	65	65	65
Ours (partial block)	42	43	35	35	35	35	67	67	67	67	67	67

Subsequent forgeries require decryption queries of 2^{14} , 2^{12} , 2^9 , 2^6 , $\dots$ and so on. Thus, for the full authentication key recovery, the required query complexity is $O(2^{\tau-d})$, and the required data complexity is $O(2^d \cdot 2^{\tau-d}) = O(2^\tau)$.

To mitigate this weakness, NIST SP 800-38D [Dwo07] heavily restricts the maximum input length and the maximum invocations of the decryption function of GCM per key for short tags, *i.e.*, $\tau = 32$ and 64. Mattsson and Westerlund [MW16] analyzed the data complexity of Ferguson’s authentication key recovery attack under these limits. As shown in Tab. 1, it ranges from 2^{62} to 2^{72} for a 32-bit tag and from 2^{79} to 2^{81} for a 64-bit tag. They are substantially larger than 2^τ , the data complexity without limits, because Ferguson’s attack must succeed in expensive subsequent forgeries multiple times after the initial forgery to recover a full authentication key. Mattsson and Westerlund [MW16] (MW) also improve the data complexity of Ferguson’s attack by taking a trade-off between the complexity of the initial forgery and that of subsequent forgeries; however, the latter remains expensive, and the resulting data complexity is still substantially larger than 2^τ as shown in Tab. 1. Although NIST recently announced it will revise NIST SP 800-38D and remove support for tags shorter than 96 bits [Nat26], GCM with short tags remains practically relevant. In particular, IPsec ESP [VM05] and IKE v2 [BM08] allow the use of GCM with a 64-bit tag. In general, short tags are attractive in bandwidth-/resource-constrained environments (e.g. a 32-bit tag is used in Zigbee, although CCM [CCM07] is used instead of GCM) or real-time communication protocols. A deeper understanding of the security implications of short tags would help evaluating the practical relevance of NIST’s revision plan.

Our contributions. We present a new authentication key recovery attack that significantly reduces the data complexity compared to Ferguson’s and MW attacks under NIST’s limits. In the best-case scenario, our attack achieves data complexity of $O(2^{\tau+1})$ for any τ , as shown in Tab. 1. Our attack improves Ferguson/MW attacks by reducing the query complexity of each expensive subsequent forgery to a single decryption query. While Ferguson/MW attacks

regenerate a completely new AD/ciphertext for each subsequent forgery, our attack derives the AD/ciphertext for subsequent forgeries from that of the initial forgery. This enables us to make use of the information obtained from the initial successful forgery, and we can construct an AD/ciphertext so that the unknown bit of the valid tag is 1. By repeating this, we can collect 1 linear equation on the authentication key with each forgery attempt after the initial forgery. This procedure of accumulating linear equations can be viewed as a generalization of bit-by-bit sub-key recovery in the universal forgery attack on GCM-SST [CMM26] proposed by Inoue *et al.* [IJMM25].

Our attack requires $2^{\tau-d+1} + (128 - \tau + d - 1)$ decryption queries with 2^d -block inputs, *i.e.*, $2^{\tau+1} + 2^d(128 - \tau + d - 1)$ data complexity in the full-block case, regardless of the NIST's limits. The query/data complexities of the initial forgery in our attack are 1 bit larger than Ferguson's to apply the bit-by-bit recovery technique. Under NIST's limits, our attack can reduce data complexity compared to Ferguson/MW as described above, but without such limits, the query/data complexities of our attack are 1 bit larger than Ferguson/MW. However, we can modify our attack for such a case by combining the Ferguson-type forgeries with the bit-by-bit recovery technique effectively. It reduces the query complexity of the third and subsequent expensive forgeries in Ferguson's attack to a single decryption query, and it may improve the constant part of the query/data complexities of Ferguson's attack.

Finally, we can also extend our attack to the nonce-misuse scenario. The resulting attack is an extension of the Joux's nonce-misuse attack on GCM [Jou06]; whereas the original attack requires full tags, our attack works even in the short-tag case by additionally performing $(128 - \tau)$ decryption queries. We will detail these points in the full version of the paper.

2 Preliminaries

For a binary string X , $|X|$ denotes its length in bits. For non-negative integer $s \leq |X|$, $\text{msb}_s(X)$ (*resp.* $\text{lsb}_s(X)$) denotes the most (*resp.* least) significant s bits of X . $X \ll s$ (*resp.* $X \gg s$) denotes the logical left (*resp.* right) shift of X by s bits. A concatenation of bit strings X and X' is denoted by $X \parallel X'$. We write 0^s to denote the s -bit string of all zeros. We may omit $\parallel$ if the notation is clear from the context.

Multiplication in GCM. The finite field used to define GCM is $\text{GF}(2^{128}) = \text{GF}(2)[x]/(f(x))$, where $f(x) = x^{128} + x^7 + x^2 + x + 1$. For a 128-bit value A , let $a_0 a_1 \cdots a_{127}$ denote the sequence of bits in A . GCM converts a 128-bit string to a polynomial in a little-endian manner; thus, A can be written as $a_0 + a_1 x + \cdots + a_{127} x^{127}$. Multiplying A by 2^{-1} (*i.e.*, x^{-1}) corresponds to a one-bit left shift $A \ll 1$ when $\text{msb}_1(A) = a_0 = 0$; otherwise, $2^{-1}A = (A \ll 1) \oplus 1^{204}10^{120}1$.

Authenticated encryption. A nonce-based Authenticated Encryption with Associated Data (AEAD) scheme $\Pi = (\Pi.\mathcal{E}, \Pi.\mathcal{D})$ is defined over a key space $\mathcal{K}$,

Algorithm $\text{GCM}.\mathcal{E}[E_K](N, A, M)$ 1. $C \leftarrow \text{GCTR}[E_K](N, M)$ 2. $T \leftarrow \text{GMAC}.\mathcal{T}[E_K](N, A, C)$ 3. return (C, T) Algorithm $\text{GCM}.\mathcal{uD}[E_K](N, A, C, T)$ 1. $M \leftarrow \text{GCTR}[E_K](N, C)$ 2. return M Algorithm $\text{GCM}.\mathcal{D}[E_K](N, A, C, T)$ 1. $b = \text{GCM}.\mathcal{V}[E_K](N, A, C, T)$ 2. if $b = \perp$ then return $\perp$ 3. else $M \leftarrow \text{GCM}.\mathcal{uD}[E_K](N, A, C, T)$ 4. return M Algorithm $\text{GCM}.\mathcal{V}[E_K](N, A, C, T)$ 1. $\hat{T} \leftarrow \text{GMAC}.\mathcal{T}[E_K](N, A, C)$ 2. if $T = \hat{T}$ then return $\top$ 3. else return $\perp$	Algorithm $\text{GCTR}[E_K](N, M)$ 1. $L \leftarrow E_K(0^n)$ 2. if $\nu = 96$ then $\text{Ctr0} \leftarrow N \parallel 0^{31}1$ 3. else then $\text{Ctr0} \leftarrow \text{GHASH}_L(\varepsilon, N)$ 4. $(M[1], \dots, M[m]) \xleftarrow{n} M$ 5. for $i = 1$ to m do 6. $C[i] \leftarrow M[i] \oplus \text{msb}_{ M[i] }(E_K(\text{inc}^i(\text{Ctr0})))$ 7. $C \leftarrow C[1] \parallel \dots \parallel C[m-1] \parallel C[m]$ 8. return C Algorithm $\text{GMAC}.\mathcal{T}[E_K](N, A, C)$ 1. $L \leftarrow E_K(0^n)$ 2. if $\nu = 96$ then $\text{Ctr0} \leftarrow N \parallel 0^{31}1$ 3. else then $\text{Ctr0} \leftarrow \text{GHASH}_L(\varepsilon, N)$ 4. $\text{Mask} \leftarrow \text{msb}_\tau(E_K(\text{Ctr0}))$ 5. $T \leftarrow \text{Mask} \oplus \text{msb}_\tau(\text{GHASH}_L(A, C))$ 6. return T Algorithm $\text{GMAC}.\mathcal{V}[E_K](N, A, C, T)$ 1. $\hat{T} \leftarrow \text{GMAC}.\mathcal{T}[E_K](N, A, C)$ 2. if $T = \hat{T}$ return $\top$ 3. else return $\perp$ Algorithm $\text{GHASH}_L(A, C)$ 1. $X \leftarrow \text{pad}_n(A) \parallel \text{pad}_n(C) \parallel \langle A \rangle_{n/2} \parallel \langle C \rangle_{n/2}$ 2. $(X[x], \dots, X[1]) \xleftarrow{n} X, Y \leftarrow 0^n$ 3. for $i = x$ to 1 do $Y \leftarrow L \cdot (Y \oplus X[i])$ 4. return Y
--	---

Fig. 1: Algorithms of $\text{GCM}[E_K]$. Note that $\nu := |N|$ and $\tau := |T|$ are fixed in advance. The operation “ $\cdot$ ” in line 3 of GHASH denotes a multiplication over $\text{GF}(2^n)$.

a nonce space $\mathcal{N}$, an associated data (AD) space $\mathcal{AD}$, a message space $\mathcal{M}$, and a tag space $\mathcal{T} = \{0, 1\}^\tau$ for some fixed tag length τ . Formally, AEAD is a tuple of an encryption function $\Pi.\mathcal{E}: \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \rightarrow \mathcal{M} \times \mathcal{T}$, and a decryption function $\Pi.\mathcal{D}: \mathcal{K} \times \mathcal{N} \times \mathcal{AD} \times \mathcal{M} \times \mathcal{T} \rightarrow \mathcal{M} \cup \{\perp\}$, where symbol $\perp$ indicates a verification failure. We require $\Pi.\mathcal{D}_K(N, A, C, T) = M$ if $\Pi.\mathcal{E}_K(N, A, M) = (C, T)$ for soundness.

3 GCM Authentication Key Recovery

3.1 Overview of our attack

Our attack has three steps. First, we obtain two valid tuples (N, A, C, T) and (N, A_1, C_1, T_1) , where $|A|, |C|, |A_1|, |C_1|$ are multiples of 128, $|A| = |A_1|, |C| = |C_1|$, and $|A|/128 + |C|/128 + 1 = 2^d$ for a positive integer d , using Ferguson’s forgery attack on GCM [Fer05]. Ferguson’s attack makes use of the difference of untruncated tags called *error polynomial*, defined as $\bigoplus_{i=1}^x (X[i] \oplus X_1[i])L^i$, where $X = A \parallel C \parallel \langle A \rangle_{64} \parallel \langle C \rangle_{64} = X[x] \parallel \dots \parallel X[1]$, and X_1 is defined analogously. By setting $X[i] \oplus X_1[i] = 0^{128}$ for i not a power of 2, the error polynomial can be

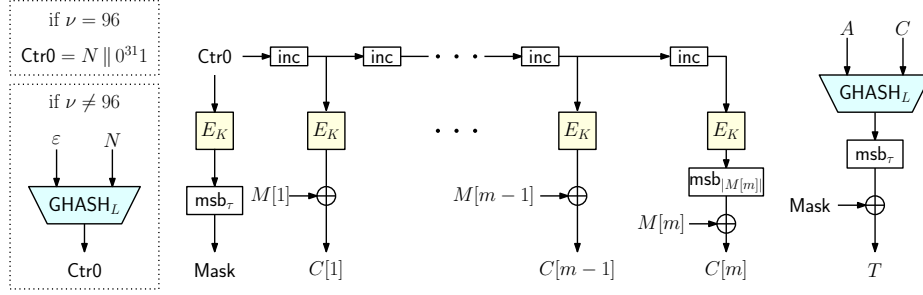


Fig. 2: Encryption of GCM.

linearized because GCM is defined in the characteristic-two field, and constant multiplication and squaring are linear. We write the linearized error polynomial as $A_D L$, where A_D is a 128×128 matrix over $\text{GF}(2)$, and we call A_D an error matrix. The purpose of this step is to obtain $T \oplus T_1$, which corresponds to the first τ -bit of $A_D L$. Ferguson's attack manipulates (A_1, C_1) so that the first $d - 1$ rows of A_D are zeros, which reduces *effective tag length* of T_1 to $\tau - d + 1$.

Second, we derive new forgeries (N, A_i, C_i, T_i) for $i = 2, 3, \dots$ from $A_D L$ used in the first step, unlike Ferguson's/MW attacks. From the result of the first step, we already know $\text{msb}_\tau(A_D L)$. We can construct (A_2, C_2) so that the corresponding error matrix $2^{-1} A_D$, and the valid tag T_2 is $T \oplus \text{msb}_\tau(2^{-1} A_D L)$. Since only one bit of $\text{msb}_\tau(2^{-1} A_D L) \approx \text{msb}_\tau(A_D L \ll 1)$ remains unknown, we obtain the correct value of T_2 in a single decryption query. The second step repeats this procedure and recovers the entire value of $A_D L$, which reveals $\text{rank}(A_D)$ linear equation on the authentication key, combined with the first step. This bit-by-bit recovery of $A_D L$ is inspired by the authentication sub-key recovery on GCM-SST [IJMM25].

Third, we recover the remaining $(128 - \text{rank}(A_D))$ -bit unknowns using the subsequent-forgery technique of Ferguson's/MW attacks. While we pointed out that the subsequent forgeries are expensive in Ferguson's/MW attacks, we have already recovered sufficient linear equations on the authentication key; thus, we can make (N, A^*, C^*, T^*) such that the effective tag length of T^* is one. By repeating this, we obtain $(128 - \text{rank}(A_D))$ linear equations on the authentication key one by one using a single decryption query.

3.2 Details of the attack

First step: initial forgery. The adversary first performs an encryption query (N, A, M) and obtains (C, T) . We assume the bit lengths of A and M/C are multiples of 128. Thus,

$$T = \text{msb}_\tau(E_K(\text{Ctr0}) \oplus \text{GHASH}_L(A, C)) = \text{msb}_\tau \left(E_K(\text{Ctr0}) \oplus \bigoplus_{i=1}^x X[i] \cdot L^i \right),$$

where $X = A \parallel C \parallel \langle |A| \rangle_{64} \parallel \langle |C| \rangle_{64} = X[x] \parallel \dots \parallel X[1]$, and $\mathbf{Ctr0}$ is a 128-bit value derived from N (see Fig. 1 for the definition). We assume $x = 2^d$ for a positive integer d . In this step, the adversary tries to make a valid forgery (N, A_1, C_1, T_1) , where $|A_1| = |A|$ and $|C_1| = |C|$. Let $X_1 = A_1 \parallel C_1 \parallel \langle |A_1| \rangle_{64} \parallel \langle |C_1| \rangle_{64} = X_1[x] \parallel \dots \parallel X_1[1]$. Since T and T_1 are valid tags, we have

$$T \oplus T_1 = \mathbf{msb}_\tau \left(\bigoplus_{i=1}^x X[i]L^i \oplus \bigoplus_{i=1}^x X_1[i]L^i \right) := \mathbf{msb}_\tau \left(\bigoplus_{i=1}^x \Delta_1[i]L^i \right).$$

We set $\Delta_1[i] = 0^n$ when i is not a power of two, and we obtain

$$T \oplus T_1 = \mathbf{msb}_\tau \left(\bigoplus_{i=1}^d D[i]L^{2^i} \right),$$

where $D[i] = \Delta_1[2^i]$. Note that $D[0] = 0^n$ because we set $|A| = |A_1|$ and $|C| = |C_1|$. As Ferguson showed [Fer05], constant multiplication and squaring are linear in $\text{GF}(2^{128})$, and the above polynomial can be linearized. Let $M_{D[i]}$ and M_S be 128×128 matrices over $\text{GF}(2)$ defined by multiplication of $D[i]$ and squaring, respectively, and we obtain

$$T \oplus T_1 = \mathbf{msb}_\tau \left(\bigoplus_{i=1}^d M_{D[i]}(M_S)^i L \right) := \mathbf{msb}_\tau (A_D L),$$

where A_D is 128×128 matrix over $\text{GF}(2)$. The adversary has $128d$ binary variables from the coefficients $D[1], \dots, D[d]$. By solving a linear system over these variables, the adversary can choose $D[i]$ so that the first $(d-1)$ rows of A_D are zero. Therefore, the first $d-1$ bits of $T \oplus T_1$ are zeros, and the effective tag length for forgery is $\tau - d + 1$. Once the adversary fixes A_D , it varies T_1 and finds valid forgery (N, A_1, C_1, T_1) , and it obtains $\tau - d + 1$ nontrivial linear equations on L . Our attack requires $(128 - \text{rank}(A_D)) \times (\tau - 1) < 128d$ so that the last step succeeds, and it is necessarily satisfied. The reason for this requirement is explained in the last step, and we explain the reason why it is satisfied in the full version of this paper. This step requires one encryption query and $2^{\tau-d+1}$ decryption queries, with each query having 2^d input blocks.

Second step: collecting more linear equations. In the previous step, the adversary obtained $\mathbf{msb}_\tau(A_D L)$. In this step, it recovers the entire value of $A_D L$ bit by bit to obtain $\text{rank}(A_D) - \tau + d - 1$ new and independent linear equations on L . To recover the $(\tau+1)$ th bit of $A_D L$, the adversary makes valid forgery (N, A_2, C_2, T_2) such that $|A_2| = |A|$, $|C_2| = |C|$. Let $X_2 = A_2 \parallel C_2 \parallel \langle |A_2| \rangle_{64} \parallel \langle |C_2| \rangle_{64} = X_2[x] \parallel \dots \parallel X_2[1]$. The adversary sets $X_2[2^i] = X[2^i] \oplus 2^{-1}D[i]$ for $i = 1, \dots, d$, and $X_2[j] = X[j]$ when j is not a power of two. The valid tag T_2 satisfies the following equation.

$$T \oplus T_2 = \mathbf{msb}_\tau \left(\bigoplus_{i=1}^d 2^{-1} D[i]L^{2^i} \right) = \mathbf{msb}_\tau(2^{-1} \cdot A_D L).$$

The adversary already knows the first τ bits of $A_D L$, and multiplying by 2^{-1} in GCM essentially shifts the value by one bit to the left. Let b be an unknown $(\tau + 1)$ th bit of $A_D L$, and we obtain

$$T \oplus T_2 = \begin{cases} (\text{lsb}_{\tau-1}(T \oplus T_1) \parallel b) & \text{when } \text{msb}_1(A_D L) = 0, \\ (\text{lsb}_{\tau-1}(T \oplus T_1) \parallel b) \oplus \text{msb}_\tau(1^2 0^4 10^{120} 1) & \text{when } \text{msb}_1(A_D L) = 1. \end{cases}$$

The adversary can obtain the correct value of b with a single decryption query; if the decryption attempt fails, it learns that the opposite bit is correct.

By repeating the above procedure, the adversary can recover unknown bits of $A_D L$ bit by bit, and obtain $\text{rank}(A_D) - \tau + d - 1$ new and independent linear equations on L . Specifically, assume that the first $\tau + s$ bits of $A_D L$ are recovered; thus, the adversary already knows the valid forgery $(N, A_{s+1}, C_{s+1}, T_{s+1})$, where $|A_{s+1}| = |A|$, $|C_{s+1}| = |C|$, $X_{s+1} = A_{s+1} \parallel C_{s+1} \parallel \langle |A_{s+1}| \rangle_{64} \parallel \langle |C_{s+1}| \rangle_{64} = X_{s+1}[x] \parallel \dots \parallel X_{s+1}[1]$, $X_{s+1}[2^i] = X[2^i] \oplus 2^{-s} D[i]$ for $i = 1, \dots, d$, and $X_{s+1}[j] = X[j]$ when j is not a power of two. It also knows $T \oplus T_{s+1} = \text{msb}_\tau \left(\bigoplus_{i=1}^d 2^{-s} D[i] L^{2^i} \right)$.

We also assume that the $(\tau + s + 1)$ th row of A_D is independent from the upper rows because otherwise, the adversary can calculate the $(\tau + s + 1)$ th bit of $A_D L$ from known $\tau + s$ bits of $A_D L$ without a query. To recover the $(\tau + s + 1)$ th bit of $A_D L$, the adversary makes (N, A_{s+2}, C_{s+2}) s.t. $|A_{s+2}| = |A|$, $|C_{s+2}| = |C|$, and $X_{s+2} = A_{s+2} \parallel C_{s+2} \parallel \langle |A_{s+2}| \rangle_{64} \parallel \langle |C_{s+2}| \rangle_{64} = X_{s+2}[x] \parallel \dots \parallel X_{s+2}[1]$, and sets $X_{s+2}[2^i] = X[2^i] \oplus 2^{-s-1} D[i]$ for $i = 1, \dots, d$, and $X_{s+2}[j] = X[j]$ when j is not a power of two. The valid tag T_{s+2} for (N, A_{s+2}, C_{s+2}) satisfies

$$\begin{aligned} & T \oplus T_{s+2} \\ &= \begin{cases} (\text{lsb}_{\tau-1}(T \oplus T_{s+1}) \parallel b) & \text{when } \text{msb}_1(T \oplus T_{s+1}) = 0, \\ (\text{lsb}_{\tau-1}(T \oplus T_{s+1}) \parallel b) \oplus \text{msb}_\tau(1^2 0^4 10^{120} 1) & \text{when } \text{msb}_1(T \oplus T_{s+1}) = 1, \end{cases} \end{aligned}$$

where b is an unknown bit derived from the unknown $(\tau + s + 1)$ th bit of $A_D L$ and the known first s bits of $A_D L$. A single decryption query recovers b and hence $(\tau + s + 1)$ th bit of $A_D L$. This step requires $\text{rank}(A_D) - \tau + d - 1$ decryption queries, with each query having 2^d input blocks.

Third step: recovering the remaining unknown bits. To recover the remaining unknown $128 - \text{rank}(A_D)$ bits of L , the adversary mounts Ferguson's forgery attack again. However, it already knows $\text{rank}(A_D)$ bits of L , and it can significantly reduce the query complexity for forgery using that information, as shown by Ferguson.

Given $\text{rank}(A_D)$ independent linear equations on L , we can write L as $L = B \cdot L' \oplus S$, where L' is a $(128 - \text{rank}(A_D))$ -bit value representing the unknown bits of L , B is a known $128 \times (128 - \text{rank}(A_D))$ matrix over $\text{GF}(2)$, and S is a known 128-bit string. Let (N, A^*, C^*, T^*) be a valid forgery that the adversary tries to make, such that $|A^*| = |A|$, $|C^*| = |C|$. Let A_{D^*} be a new 128×128 bit error matrix defined from the difference between (A, C) and (A^*, C^*) , and we

obtain

$$\begin{aligned} T \oplus T^* &= \text{msb}_\tau(A_{D^*}L) = \text{msb}_\tau(A_{D^*}(B \cdot L' \oplus S)) \\ &= \text{msb}_\tau(A_{D^*}BL' \oplus A_{D^*}S). \end{aligned}$$

In the same manner as the first forgery, the adversary can choose A^* and C^* so that the first $\tau - 1$ rows of $A_{D^*}B$ are zero, because we set $(128 - \text{rank}(A_D)) \times (\tau - 1) < 128d$. Therefore, the effective tag length of T^* is one, and the adversary obtains a linear equation over L' with a single decryption query. After that, it can either recover the entire value of $A_{D^*}BL'$ bit by bit, or create a new error matrix and obtain one new linear equation of L . In either case, this step requires $128 - \text{rank}(A_D)$ decryption queries, with each query having 2^d input blocks.

3.3 Complexity analysis and comparison

We present the complexity analysis of our attack described in Sect. 3.2 and compare it with Ferguson/MW attacks under NIST's limits. Note that the analysis and comparison are based on the best-case scenario for each attack. Ferguson/MW attacks assume that the original input is *partial block*; the original AD or ciphertext is not a multiple of 128. Our attack described in Sect. 3.2 assumes that the original input is *full block*; the original AD and ciphertext are multiples of 128.

The number of decryption queries and data complexity of our attack described in Sect. 3.2 are as follows.

$$\begin{aligned} \text{Query: } & 2^{\tau-d+1} + (128 - \tau + d - 1) \\ \text{Data: } & 2^{\tau+1} + 2^d(128 - \tau + d - 1) \end{aligned}$$

For the 32-bit and 64-bit tag cases, data complexity is 2^{33} and 2^{65} , respectively, under NIST's limits. This is because our attack requires only multiple forgeries with an effective tag length of one bit after the initial forgery, and its query complexity $(128 - \tau + d - 1)$ is lower than NIST's limits. On the other hand, Ferguson/MW attacks require *expensive* subsequent forgeries (*i.e.*, whose query complexity exceeds NIST's limits) after the initial forgery. For example, when $\tau = 32$, $\log_2 \ell = d = 6$, and $\log_2 q = 11$, Ferguson/MW attacks require 5 successful forgeries, and their complexities are as follows. We also add the complexities of our attack for comparison.

$$\begin{aligned} \text{Ferguson: } & 2^{26}, 2^{25}, 2^{23}, 2^{18}, 2^{11} \\ \text{MW: } & 2^{32}, 2^{24}, 2^{22}, 2^{17}, 2^9 \\ \text{Ours: } & 2^{27}, \underbrace{1, 1, \dots, 1}_{101} \end{aligned}$$

Whereas the complexity of the initial forgery in our attack is one bit larger than that of Ferguson's attack, the total data complexity of our attack is much smaller than that of the Ferguson/MW attacks under NIST's limits, thanks to the subsequent one-bit forgeries as shown in Tab. 1.

Table 2: Decryption query and data complexities required for the initial forgery under NIST’s limits.

	query	data
Ferguson [Fer05]	$2^{\tau-d}$	2^τ
MW [MW16]	2^τ	2^τ
Ours (full block)	$2^{\tau-d+1}$	$2^{\tau+1}$

Increase of complexity for the initial forgery. In Ferguson’s (*resp.* MW) attack, the query and data complexity required for the initial forgery is $2^{\tau-d}$ and 2^τ (*resp.* 2^τ and 2^τ) in the best-case scenario, which is one bit smaller than our attack, as shown in Tab. 2. This is because Ferguson/MW attacks can set $D[0] := \langle |A| \rangle_{64} \parallel \langle |C| \rangle_{64} \oplus \langle |A_1| \rangle_{64} \parallel \langle |C_1| \rangle_{64} \neq 0^{128}$ without modifying the number of input blocks when the input block is partial. They can make the first d rows of A_D zero by manipulating $D[1], \dots, D[d]$, not $d-1$ rows. Our attack basically cannot do that because it repeatedly multiplies the error polynomial by 2^{-1} in the second step, and it is difficult to maintain the consistency between the non-zero $2^{-s}D[0]$ and other blocks.

3.4 Application to the partial-block case

When the length of original AD or ciphertext is not a multiple of 128, the query complexity of our attack may increase. Our attack requires the lengths of the ADs/ciphertexts in the original and the initial forgery to be the same, as described in Sect. 3.3. If the partial blocks correspond to $D[\cdot]$, the adversary basically cannot set the desired difference to the block; thus, such blocks are unusable. The extent to which the complexity increases depends on d .

When $d \geq 3$, the adversary can use at least $(d-2) D[\cdot]$; thus, it can mount the attack in Sect. 3.2 using that block. This results in increasing the query complexity of the initial forgery by at most 2 bits. When $d \leq 2$, if the adversary can use at least one $D[\cdot]$, it can mount the attack in the case of $d \geq 3$. Otherwise, it cannot use any $D[\cdot]$, and the attack in Sect. 3.2 cannot be applied directly. In such a case, our attack can be modified as follows. In the first step, the adversary performs two Ferguson-type forgeries by modifying $D[0]$ so that the inputs of the initial and second forgeries are full blocks and have the same length. As a result, the adversary obtains valid tuples (N, A_1, C_1, T_1) and (N, A_2, C_2, T_2) , where inputs are full blocks and $|A_1| = |A_2|$, $|C_1| = |C_2|$. It can then return to the second step of the attack in Sect. 3.2, by multiplying $T_1 \oplus T_2$ by 2^{-1} . The above modified attack requires an expensive second forgery, which somewhat increases data complexity compared to the case of $d \geq 3$ under NIST’s limits, as shown in Tab. 1.

References

- BM08. D. Black and D. McGrew. Using Authenticated Encryption Algorithms with the Encrypted Payload of the Internet Key Exchange version 2 (IKEv2) Protocol. RFC 5282, 2008. Accessed: 2026-07-23.
- CCM07. Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality. NIST Special Publication 800-38C, 2007. National Institute of Standards and Technology.
- CMM26. Matt Campagna, Alexander Maximov, and John Preuß Mattsson. Galois Counter Mode with Strong Secure Tags (GCM-SST). Internet-Draft draft-mattsson-cfrg-aes-gcm-sst-21, Internet Engineering Task Force, July 2026. Work in Progress.
- Dwo07. Morris Dworkin. *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. NIST-SP 800-38D, 2007.
- Fer05. Niels Ferguson. Authentication Weaknesses in GCM. Public Comments to NIST, 2005. <http://csrc.nist.gov/groups/ST/toolkit/BCM/comments.html>.
- IJMM25. Akiko Inoue, Ashwin Jha, Bart Mennink, and Kazuhiko Minematsu. Generic security of GCM-SST. In Marc Fischlin and Veelasha Moonsamy, editors, *ACNS 2025, Part II*, volume 15826 of *LNCS*, pages 342–369. Springer, Cham, June 2025.
- Jou06. Antoine Joux. Authentication Failures in NIST version of GCM. Public Comments to NIST, 2006. <http://csrc.nist.gov/groups/ST/toolkit/BCM/comments.html>.
- MW16. John Mattsson and Magnus Westerlund. Authentication key recovery on Galois/counter mode (GCM). In David Pointcheval, Abderrahmane Nitaj, and Tajjeeddine Rachidi, editors, *AFRICACRYPT 16*, volume 9646 of *LNCS*, pages 127–143. Springer, Cham, April 2016.
- Nat26. National Institute of Standards and Technology. NIST to Revise Special Publication 800-38D, 2026. <https://csrc.nist.gov/pubs/sp/800/38/d/r1/2prd>.
- VM05. John Viega and David McGrew. The Use of Galois/Counter Mode (GCM) in IPsec Encapsulating Security Payload (ESP). RFC 4106, 2005. Accessed: 2026-07-23.

14. Thales (Graham COSTA), July 31, 2026

Dear NIST,

On behalf of Thales, I would like to submit responses to questions 1a and 1b from your second pre-draft call for comments.

On 1a: Reduction of Supported Tag Length Options

For the products we reviewed internally, we typically support only 96-bit or 128-bit tags. As a result, we do not anticipate significant challenges from reducing the set of supported MAC tag lengths. Such a change would simplify implementations and reduce the burden on vendors offering general-purpose cryptography-as-a-service solutions by limiting the range of tags they must support.

That said, consistent with the guidance and examples in Appendix C of the current standard, we are aware of established protocols that use shorter tags, including SRTP and cellular link-layer protocols used in 3GPP and 5G standards.

On this basis, NIST may wish to consider an initial step in SP 800-38Dr1 that deprecates, but does not yet disallow, shorter tags, with their eventual removal deferred to a future revision.

An alternative approach would be to continue permitting shorter tags for specific protocols identified by NIST. This could be limited to protocols that can demonstrate compliance with the criteria for short tags described in Appendix C of the current SP 800-38D.

We also note that newer protocols such as COSE (RFC 9052 and RFC 9053) recommend CCM as an alternative authenticated-encryption mode when shorter tags are required. COSE otherwise mandates 128-bit tags when using GCM. Guidance on the trade-offs between these two modes in a new appendix may help address concerns arising from a significant reduction in supported tag lengths in the next revision of SP 800-38D.

On 1b: Variable-Length Tags

We do not see a reason to retain support for variable-length tags outside a predefined set of options (for example, 96-bit and 128-bit tags).

Other Comments:

Linked to comments above on the use of short tags for 5G and adoption of GCM-SST, we are also aware of ongoing efforts to standardize this through the IETF (draft-mattson-cfrg-aes-gcm-sst-21). This will likely eventually be established as another spur from GCM looking to address

known shortfalls (or risks) associated with the standard GCM mode. The other established related mode being AES-GCM-SIV as standardized in RFC 8452.

Alongside considering reduction in tag lengths, we would separately encourage NIST to consider other established community driven variants such as GCM-SST and GCM-SIV for adoption in SP 800-38Dr1.

Lack of support for modes such as GCM-SST and GCM-SIV, despite broader cryptographic community adoption to address concerns with GCM, forces users to use otherwise FIPS 140 validated cryptographic modules in non-approved modes of operation.

Thank you again for the opportunity to provide feedback on your standards. Please do not hesitate to contact us if you have any follow-up questions.

Kind Regards,

Graham COSTA (he/him)

Security and Certifications Manager

Cyber Security Products / Cybersecurity and Digital Identity / UK

Thales

15. Google (Victor Escobedo), July 31, 2026

Thank you for the opportunity to provide feedback on the Second Pre-Draft Call for Comments for SP 800-38D Rev. 1 regarding the GCM and GMAC Block Cipher Modes of Operation. Please find our formal comments and responses to the relevant questions below.

Questions on GCM

Question 1a: The revision to GCM will remove support for authentication tags whose lengths are less than 96 bits. Which of the remaining tag sizes (i.e., 96, 104, 112, 120, or 128 bits) are needed for GCM? In particular, would tag sizes of 96 bits and 128 bits be sufficient for applications?

Yes, retaining only the 96-bit and 128-bit tag sizes would be sufficient for our applications. In practice, our implementations almost exclusively utilize 128-bit authentication tags.

Question 1b: GCM currently supports variable-length IVs by applying the internal hash function to generate a new, 96-bit IV value. Are variable-length IVs important for any applications?

Variable-length IVs are not critical for our applications. We currently restrict our support to 96-bit IVs, primarily because hashed IVs present a more complex collision analysis and generally result in a reduced limit on the allowable number of invocations.

Questions on wGCM

Question 2a: Would a limit of 264 wGCM invocations per key be sufficient?

Yes, a limit of 264 wGCM invocations per key would be entirely sufficient for our current and anticipated use cases.

Question 2b: What authentication tag sizes should be specified for wGCM (e.g., 128, 192, or 256 bits) to support larger data limits than GCM? Is more than one tag size necessary?

We recommend retaining the existing GMAC implementation structure without modification and specifying a single 128-bit authentication tag size for wGCM. We do not believe more than one tag size is necessary to support our applications.

Question 2g: The natural generalization of the GCM hash function (i.e., GHASH) for wGCM would specify a suitable polynomial of degree 256 to define a hash function (called wide-GHASH)... Should NIST consider a different option for the hash function component of wGCM?

We recommend retaining the existing GHASH component without modification rather than introducing wide-GHASH or concat-GHASH. The primary limitation in current GCM deployments stems from the GCTR component, where the 96-bit IV limits usage to 232 invocations before the probability of IV reuse exceeds 2^{-32} , thereby risking the leakage of the GHASH authentication key.

Fixing this IV restriction does not require expanding the hash function. Given the super-linear computational cost associated with increasing the size of the GHASH field, we advise leaving GHASH as it currently stands and focusing any modifications strictly on the AES/GCTR components of AES-GCM.

Furthermore, we strongly encourage NIST to consider adopting AES-GCM-SIV as a separate Special Publication. AES-GCM-SIV provides a robust, standardized solution to the IV reuse vulnerability and is significantly easier to implement securely in practice.

Sincerely,

Victor Escobedo

US Standards Lead, Google

Dear NIST,

Thank you for providing the opportunity to provide feedback on the revision of Special Publication (SP) 800-38D, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC.

The comments below have been collected and provided by the NSA Center for Cybersecurity Standards. The feedback represents comments from SMEs across the NSA.

We are happy NIST continues to look at improved modes. While wGCM addresses some security issues, it still suffers from the “forbidden attack” by Joux. Our responses follow below -

1. Questions on GCM:

- a. The revision to GCM will remove support for authentication tags whose lengths are less than 96 bits. Which of the remaining tag sizes (i.e., 96, 104, 112, 120, or 128 bits) are needed for GCM? In particular, would tag sizes of 96 bits and 128 bits be sufficient for applications?

Tag sizes of 96 and 128 bits should be sufficient. The security world has largely aligned around 128-bit tags for clean 64-bit architecture efficiency (and increased security), or dropped down to 96 bits exclusively for network routing headers.

- b. GCM currently supports variable-length IVs by applying the internal hash function to generate a new, 96-bit IV value. Are variable-length IVs important for any applications?

We do use variable length IVs in some applications, but a fixed size IV is much simpler and we recommend its use.

2. Questions on wGCM:

- a. Would a limit of 2^{64} wGCM invocations per key be sufficient?

While the 2^{64} limit should be sufficient for most applications, this may pose an issue in the not-so-distant future for some. A higher limit should be considered; for example, if using a random 192-bit IV construction with no free field, one can accommodate 2^{80} invocations and still achieve a 2^{-32} probability of invocation with the same IV and the same key on two different inputs. As 2^{-32} is current practice and is sufficient for GCM, we see no reason to lower it further.

- b. What authentication tag sizes should be specified for wGCM (e.g., 128, 192, or 256 bits) to support larger data limits than GCM? Is more than one tag size necessary?

Smaller (truncated) tags are typically used to balance network efficiency and hardware constraints against security requirements (forgery resistance). However, the forgery resistance should not dip below that which is currently provided by GCM with 96-bit tags. With this in mind, tag sizes of 256 and 192 should be allowed, and 128 bits if wide-GHASH is used.

- c. NIST proposes 192 bits as the default IV length for wGCM, devoting the remaining 64 bits of the IV block within wGCM to the message block counter for the GCTR encryption. Is there a need to support longer IVs in wGCM by applying the internal hash function, as in GCM?

A fixed 192-bit IV length is large enough to be chosen completely at random without a meaningful risk of collision and is sufficient for our use cases.

- d. The deterministic IV construction for GCM defines a fixed field and an invocation field. For wGCM, NIST proposes to recommend a fixed field of 96 bits and an invocation field of 96 bits to promote interoperability. Are other options needed?

Consider aligning the size of the invocation field with the allowed number of invocations. Regardless, a 96-bit invocation field for the deterministic IV construction is fine.

- e. The random IV construction for GCM defines a random field and a free field. For wGCM, if the number of invocations per key were limited to 2^{64} , and if 192 bits were specified for the random field, then the probability of an IV collision would be limited to 2^{-64} . Is the free field useful/necessary for wGCM?

The free field in the random IV construction seems unnecessary. Many standard implementations of GCM use a 96-bit random field with a 0-bit free field, as it provides optimal security bounds without additional complexity. The free field should only be considered if variable length IVs are allowed.

- f. Are these two 192-bit IV constructions sufficiently flexible, or should NIST consider adding other, comparably secure constructions?

These two IV constructions cover typical ways data is transmitted: in sequential states (like network packets) or isolated instances (like stateless API calls). Thus, any alternative method would likely be a redundant variation of these two. Limiting the official standard to exactly these two constructions would also help to minimize human implementation errors.

- g. The natural generalization of the GCM hash function (i.e., GHASH) for wGCM would specify a suitable polynomial of degree 256 to define a hash function (called wide-GHASH) on 256-bit blocks. The output can be truncated as needed to obtain the appropriate tag size. Should NIST consider a different option for the hash function component of wGCM? For example, two instances of GHASH using independent

128-bit keys could be concatenated and truncated as needed. Since this option (called concat-GHASH) operates on 128-bit blocks, it would be more efficient than wide-GHASH. However, concat-GHASH cannot provide as much generic security assurance as wide-GHASH and would require much more restrictive usage bounds (i.e., smaller limits on total data, the message length, and invocations per key) for any given security target.

No, we feel that wide-GHASH should be used as it offers the best security properties. While the concat-GHASH option is more efficient, an unfortunate issue with GHASH is that it loses k bits of forgery resistance when used to authenticate 2^k blocks of data [1]. This fact is often missed by engineers, and as concat-GHASH loses $2k$ bits of forgery resistance, we recommend against using it.

[1] Niels Ferguson, Authentication weaknesses in GCM, 2005.

Thank you for your consideration.

Should you have any questions, please contact Mike Boyle (cc'ed) and myself.

Regards,
Denisha Warren

Denisha Jackson Warren
Center for Cybersecurity Standards
Cybersecurity Collaboration Center
National Security Agency

