Practical Challenges with AES-GCM and the need for a new cipher

Panos Kampanakis
kpanos@amazon.com

Matt Campagna
campagna@amazon.com

Eric Crocket
ericcro@amazon.com

Adam Petcher
apetcher@amazon.com

Shay Gueron
sgueron@univ.haifa.ac.il

March 22, 2024

1 Introduction

AWS is pleased to see the Third NIST Workshop on Block Cipher Modes of Operation 2023 which plans to address limitations of block cipher modes (SP 800-38 series). We also welcome NIST's interest in discussing wide-block encryption techniques.

AWS has a few cloud encryption use-cases that could benefit from efficient new wide-block ciphers and new quantum-safe key transport techniques. This submission presents some of these use-cases and tries to explain the challenges in detail. In summary, as suggested in AWS' feedback on FIPS 197 [5], AWS would like to see standardized a **new block cipher and an Authenticated Encryption with Additional Data (AEAD) mode** with the following properties:

- 256-bit block width
- efficiency (better than or equivalent to AES-GCM)
- ability to encrypt (at least) 2^{64} or (ideally) 2^{96} messages with random Initialization Vectors (IV). This could be achieved with a (minimum) 192-bit or (ideally) 256-bit IV length or other method.
- a key / context commitment option for robustness if we can assume acceptable performance overhead.
- a nonce misuse-resistance option. This would be a nice-to-have feature, but not mandatory, especially if we can send a 2^{64} or 2^{96} messages with random IVs.

- streaming support which allows for starting encrypting/decrypting without holding the whole message in memory.
- block cipher invocation which doesn't depend on the plaintext so that we can compute the keystream in advance

AWS would also like see standardized a **new asymmetric, quantum-safe key-wrapping primitive** that adds on the asymmetric key transport techniques in SP 800-56B and will be FIPS-approved.

2 New Block Cipher

2.1 AES-GCM Pain-Points

AWS has been using AES-GCM in many encryption use-cases. AES-GCM has good performance, is highly optimized, FIPS-approved, and well-trusted. In spite of its proven value, AWS use-cases sometimes face challenges with this mode which we would like to have addressed. Some of them are discussed below.

2.1.1 Limitations of Random IVs

Recall that reusing an initialization vector (IV) with the same key with AES-GCM (using any IV construction) results in a loss of security guarantees, and is generally considered a"catastrophic failure". With random IVs in AES-GCM, NIST limits the number of messages that can be encrypted with a single key to 2^{32}, which ensures that the probability of a {key, IV} collision is less than 2^{-32}.

The issue becomes worse if we consider random IVs in the multi-user setting. Multi-user security considers how the security of a system degrades with the use of multiple keys. One setting where this arises is in systems with a large number of users, each with one key, hence the name "multi-user security". However, the same situation exists in many AWS systems without this classic definition of "users". For example, many AWS systems use multiple keys, rotate keys, or derive multiple keys. Consider a system that uses a single key which is used to encrypt 2^{32} messages. This system has an IV reuse probability $\leq 2^{-32}$, which is consistent with NIST's requirements. However, a system that uses 128 keys where each key is used to encrypt 2^{32} messages has an IV reuse probability of 2^{-26}. Put another way, in a system that uses 128 keys, each key should be used to encrypt no more than 2^{29} messages to ensure that the overall probability of IV reuse in the system remains $\leq 2^{-32}$.

There are IV-reuse resistant modes like AES-GCM-SIV [14],GCM-SIV [15], but these are a) not FIPS approved, and b) less efficient. Prior to modes, in 2013, we developed a derived-key mode for AWS Key Management System (KMS) to comply with both FIPS requirements and scale the use of GCM – we published this years later [6].

Some challenges with random IVs in AWS use-cases follow below:

Random IVs for transport encryption: For many AWS use-cases, the 2^{32} limit imposes the burden of frequent re-keying. While we automate key rotation when possible, extremely short rotation periods impose operational risk. For example, we have a use-case involving a large number of encryption devices that share a single key. These devices collectively encrypt 2^{32} messages in less than two seconds. Any key rotation outage in this scenario could result in catastrophic IV reuse, so this use-case is not able to take advantage of a stateless random IV construction.

Random IVs in KMS: GCM is an attractive mode to use for a service like AWS KMS, where a customer managed key (CMK) is only accessible within a FIPS 140-2 Level 3 HSM. The CMK is used to both encrypt/decrypt small plaintext values and request encrypted data keys. GCM is a convenient mode for both operations. Constraints on the system, limit the number of CMKs we can manage in or even export out of the HSM. AWS KMS, like many cloud services, must deliver high data transaction rates over a distributed system of variable size fleets (hosts of the same functionality). For these reasons, the state management required to use a deterministic IV over a volatile fleet of HSMs is not feasible. It would not be unreasonable for some customers to exceed 2^{32} calls under a CMK within weeks (fig. 1). Rotating significantly faster than once a week, results in a high-volume of keys to be managed by the service. AWS KMS compensates for this by using a derived key mode to encrypt requests to CMKs [6]. This increases the size of the resulting ciphertext and increases the cryptographic operations to deliver a core functionality of AWS KMS.

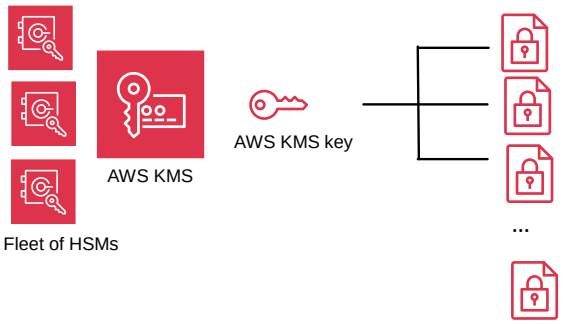

Figure 1: A KMS key could be used to encrypt way more than 2^{32} plaintexts in a week for some customers.

A standardized 256-bit block cipher and commensurate GCM mode, that accommodated larger nonces, would alleviate such issues. It could provide us with at least 2^{64} encryptions with a probability of a {key, IV} collision of 2^{-64} and up to 2^{112} encryptions before reaching a probability of 2^{-32}.

2.1.2 Limitations of Counter / Deterministic IVs

In practice, AWS systems frequently use lots of keys over their lifetime, hence random IVs with a $\leq 2^{-32}$ reuse probability are not an option. In such distributed environments, AES-GCM may use a deterministically generated IV which increments with every invocation (as a counter). Deterministic IVs with AES-GCM come with their own limitations and problems, including the following:

1. There is insufficient room in a 96-bit IV for a unique fixed field and a counter big enough to support large use-cases. A general extension to the network encryption system described above requires additional contextual information to disambiguate devices. The network encryption system uses 32-bit field as device IDs with a 64-bit counter, so this extended system has to reduce the counter size to account for a larger fixed field, reducing the number of messages that can be encrypted with a single key. Another problem with this system is that it requires a *separate system* to ensure unique device identifiers. We would prefer to use random device identifiers for this use-case, but we would need at least 96 bits of fixed field to do so, leaving no room for a counter. A similar problem arises in large auto-scaling distributed systems, in which an unbounded number of processes share an encryption key. Partitioning the counter space and assigning a prefix to each process is challenging because the number of processes that will be created over the lifetime of a key can be very large.

2. Ensuring a counter is not reused across system reboots and resets is challenging, especially when trying to meet the FIPS requirements about IV uniqueness proof or reuse checks, zeroization, and more. This requires non-volatile memory to store the last IV. Using random IVs would dramatically simplify the FIPS certification process because we would not need to maintain state on these devices.

3. When the key and counter are shared in a distributed system, it is difficult to develop a trustworthy and efficient mechanism to increment the counter.

A standardized 256-bit block cipher with large nonces would mean we don't need to rely on the deterministic IV constructions to protect against {key, IV} reuse for many applications, removing the need to maintain state on the last used IV.

2.1.3 PRP Limits

A 128-bit block cipher like AES-GCM is a Pseudo Random Permutation (PRP) of $\{0,1\}^{128}$ regardless of the key length. Its output is distinguishable from random (with high-enough probability) after 2^{64} invocations according to NIST (Appendix B of SP800-38D). draft-irtf-cfrg-aead-limits provides formulas for calculating the security of AES-GCM based on different parameters. Using these formulas, TLS 1.3 (i.e., RFC8446, Section 5.5) recommends encrypting $2^{34.5}$ blocks maximum per key for 2^{-57} Authenticated Encryption (AE) security.

Security-wise, this limits the number of calls allowed with a given key. At cloud scale this quickly becomes a cause for re-keying as in the examples below.

Distributed transport encryption: This use-case involves a large number of devices sharing a single key. These devices collectively encrypt 2^{64} blocks every two weeks (fig. 2). Maintaining a shared key across this large, diverse fleet is operationally challenging. Additionally, having a two-week key lifetime adds significant operational risk, because if key rotation fails, a scenario which is not uncommon, engineers are racing against a clock to get the problem fixed before the two-week limit. One possible solution is to distribute multiple keys at once, however this reduces the perfect forward secrecy of the system.

Figure 2: High-volume transport encryption could collectively encrypt 2^{32} messages in seconds and 2^{64} in weeks.

High-Speed Optical Transport Network Encryption: Flexible Optical Transport Network (FlexO) is a set of transport technologies developed in the ITU. Optical Transport Network (OTN) can operate at very high speeds (e.g., 500Gbps) which impose challenges on the use of AES-GCM. OTN may use large frames (e.g., 80KB) and have very low latency requirements. Operating and authenticating such big frames over 128-bit blocks could lead to unacceptable latency. For example OTNsec provides encryption for FlexO. It uses the IKEv2 protocol to negotiate and establish the cryptographic Security Associations and then encrypts the frames using the negotiated block cipher, usually AES-GCM. 2^{32} maximum total invocations for an OTN of 500Gbps with a 80KB frame size leads to a 91 minute rekey frequency with random AES-GCM IVs. With minimum size frames like 2KB, the rekey time would be 3 seconds. To address this, GCM-AES-XPN uses deterministic IVs which means that rekeys do not need to happen after 2^{32} invocations [19]. The challenge in this scenario for 80KB frames is that AES-GCM may not perform at these very high speeds for 5,000 blocks. There have been attempts to address these challenges by introducing wide-block ciphers like [1].

OTNsec and similar high speed transport encryption use-cases could use a standardized 256-bit block cipher or a new and efficient block mode.

2.1.4 Key / Context Commitment

It has been demonstrated in many real world applications, key or context committing modes would enhance the security of applications [8], [17] and [2]. A lack of key commitment allows for a sender to craft a single authenticated ciphertext that decrypts to two plaintexts, potentially confusing recipients (fig. 3). The AWS Encryption SDK was updated in 2020 to include key commitment functionality.

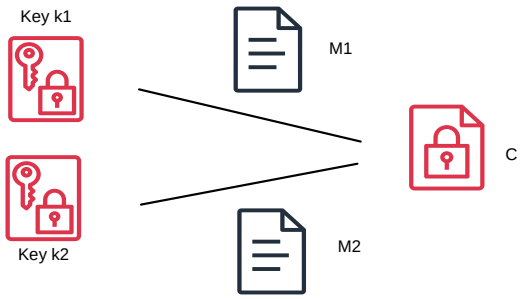

Figure 3: Two keys used to encrypt two plaintexts without key commitment could lead to the same ciphertext which means decryption could lead to different results depending on the key used.

Security and software engineers would benefit from a NIST-approved key or context committing block cipher mode. For more details, refer to [18] and [2].

2.1.5 Short Tags

The probability of selected forgery for GCM is $n/2^t$ where n is the number of blocks of the AAD and ciphertext, and t is the bit length of the tag [11]. This combined with leakage of the H value results in fairly severe requirements for short tags in [9]. This makes GCM unattractive for some cellular and other bandwidth constrained communication channels.

2.2 Discussion about XTS

AES-XTS provides adequate security margins for disk encryption in typical scenarios where the adversary is only allowed to view the ciphertext once. This section describes some theoretical benefits of using a wide-block PRP with XTS mode, but there are no significant practical benefits over AES-XTS.

XTS could be standardized for use with a wide-block PRP, and this would enable better theoretical security margins in circumstances where the adversary is allowed to view multiple different ciphertexts encrypted under the same key. In these circumstances, a wider PRP would reduce the probability of a collision which would allow the adversary to learn the initial tweak value for an XTS data unit. An upper bound on the probability of such a collision after encrypting a maximum of 2^{20} blocks with AES-XTS as mandated by [10] is 2^{-88}. A 256-bit

PRP would improve the collision probability (to $1 - e^{-1/2^{217}}$) but AES-XTS' margins are adequate for up to 2^{20}-block plaintexts. A wide-block XTS would also enable larger data units so that initial tweak values must be calculated less frequently. But this would not have a significant impact on performance because the maximum data unit size for AES-XTS allows an initial tweak to be calculated once every 2^{20} AES blocks.

2.3 Need for a new wide-block cipher and mode

It is clear that some use-cases could benefit from a new block cipher with the following properties:

- 256-bit block width
- efficiency (better or equivalent than AES-GCM)
- ability to encrypt (at least) 2^{64} or (ideally) 2^{96} messages with random Initialization Vectors (IV). This could be achieved with a (minimum) 192-bit or (ideally) 256-bit IV length or other method.
- a key / context commitment [13] option for robustness if we can assume acceptable performance overhead.
- a nonce misuse-resistance option [15, 14] as a nice-to-have feature, but not mandatory.
- support streaming which allows for starting encrypting/decrypting without holding the whole message in memory.

Below we brainstorm potential options for a 256-bit cipher and a block mode that satisfy these requirements. We do not come up with concrete solutions, but we share some thoughts and alternatives to consider. Literature and other potential options ought to be carefully evaluated.

2.3.1 New wide-block cipher

We need a new wide-block cipher built from Pseudorandom Functions (PRFs). The main benefits of such ciphers are:

- We can quickly deploy (somewhat) efficient implementations using hardware acceleration that is already present on modern CPUs.
- Standardizing these ciphers and producing more efficient hardware for them should be relatively simple, because we are using existing building blocks.
- We can argue that all of this is secure using existing proofs/standards.

An AES-based primitive is preferrable so we can reap the optimizations of AES instruction sets and leverage the investment of the ecosystem in hardware and software support for some components of the AES, specifically, inversion in $GF(2^8)$. This operation is the major element that speeds the instructions and the full AES (and Rijndael-256) implementation.

A straightforward 256-bit block candidate cipher would be Rijndael-256, which was part of the original proposal for AES (although only the 128-bit block size was standardized). Code that executes Rijndael-256 could use the AES-NI instruction set ([12, Fig 30]). In 2016, the throughput of code that uses AES-NI, running pipelined Rijndael-256 encryption, was 1.54 cycles per byte (cpb) [16]. By comparison, AES throughput on such processors is 0.65 cpb. Recent architectures have doubled and quadrupled the throughput of AES and also offer instructions to execute $GF(2^8)$ directly, which can be used for AES, Rijndael-256 and other constructions. Hardware implementation of Rijndael-256 could reuse components that support AES.

There are also some alternatives that rely on the AES round as a "fast primitive", and may constitute a different block cipher, even with flexible block sizes [16]. As an example, Simpira 256-bit permutation has throughput of 0.94 cpb, and it can be used e.g., with an Even-Mansour construction, to define a block cipher.

Other options for wide-block ciphers include PRFs based on HMAC like HMAC-CTR as a simple nonce-based IND-CPA encryption scheme.

2.3.2 New Mode

Given a PRP with large (e.g. 256-bit) domain and range, we can construct an authenticated encryption scheme using the OCB mode of operation. OCB mode provides authenticated encryption from the sole assumption that the underlying encryption algorithm is a PRP, so it can easily be adapted for use with new encryption algorithms that are developed in the future. In contrast, modes like GCM would also require the development of a separate authenticator to be paired with the encryption algorithm. OCB is very efficient when paired with AES on modern CPUs, and the standardization of OCB has the potential to immediately improve performance for current workloads on some CPUs. On Graviton3 CPUs, AES-128/OCB is potentially 50% faster than AES-128-GCM.

Stream cipher constructions are another viable approach for providing authenticated encryption, assuming the security margins that follow from nonce sizes and proof bounds are significantly better than those in AES-GCM. When the construction is based on a 128-bit PRP, it provides insufficient security margins due to the potential for random IV collisions and the lack of collisions in PRP output, as described in section 2.1. Fortunately, we can obtain acceptable margins by replacing the 128-bit PRP with a 256-bit PRP and allowing larger IVs.

It is necessary to pair a stream cipher with an authenticator to obtain authenticated encryption. Developing a new Carter-Wegman authenticator is a reasonable option, but practical experience has shown that this authenticator

is brittle, and that it provides insufficient security in circumstances where the tag is truncated or robust encryption is expected. Another option is to pair the stream cipher with an authenticator based on a collision-resistant PRF such as HMAC/SHA-256, which would enable tag truncation and robust encryption. A collision-resistant PRF will likely be much slower than a Carter-Wegman authenticator, so perhaps this should be an option that can be employed when it is needed.

It is possible to produce the keystream using existing primitives that are already accelerated on modern CPUs. For example, AEGIS-128L [7] is an AES-based AEAD algorithm designed for performance. As another example, an encryption mode (without authentication) using HMAC/SHA-256 in counter mode can be approximately ⅕ the speed of AES-128-CTR on Graviton3 CPUs. This sort of mode could be used as a basis for judging the performance of other wide-block modes and primitives.

2.3.3 Nonce-misuse resistance

Nonce-misuse-resistant modes of operation are inherently slower during encryption than typical nonce-based encryption modes due to the requirement to make two passes over the plaintext. They are also not FIPS-approved. Ideally, standards would provide options for nonce-misuse-resistant modes in circumstances where this decreased performance is tolerable.

One possibility is to standardize a general-purpose SIV construction that can be used with any authenticated encryption mode. For example, the IV could be produced by applying a PRF to the plaintext. The main drawback of this sort of construction is that it can be inefficient when used with some authenticated encryption modes. For example, AES-GCM would also process the ciphertext using the authenticator, and this "extra" processing could be eliminated by using a specialized SIV mode like AES-GCM-SIV [15, 14] . For highly-optimized modes like OCB, the cost of this "extra" processing is negligible, and a general-purpose SIV construction would be approximately as efficient as a more specialized SIV mode.

3 New asymmetric, quantum-safe key-wrapping primitive

Deviating for the block symmetric encryption topic, RSA has been widely used for key transport and envelope encryption in protocols and cryptographic use-cases like AWS KMS. Since NIST is working on new post-quantum primitives, we want to bring up the need for a new asymmetric, quantum-safe key transport primitive that adds on the quantum-vulnerable key transport techniques in SP 800-56B and will be FIPS-approved.

New quantum-safe Key Encapsulation Mechanisms being standardized by NIST can't be used the same way as RSA to encrypt short plaintexts for key transport or envelope encryption. One approach to achieve that is to extend

HPKE [4] to use post-quantum hybrid key estabishment [3, 20]. HPKE [4] "base" encryption has been proven to be CCA2 secure, and post-quantum HPKE can be proven CCA2 secure as well [3, §IV-C]. HPKE also uses NIST approved primitives like ECDH, HKDF key derivation and thus can be FIPS-approved.

NIST should standardize a quantum-safe key transport mechanism to replace RSA in SP 800-56B soon. HPKE is a good candidate.

4 Conclusion

In conclusion, we discussed some of the pain points AWS is facing with AES-GCM and stressed the need for a new 256-bit block cipher and a mode of encryption which is efficient and has certain properties. We also suggested the standardization of a quantum-safe key transport mechanism.

References

[1] Farzaneh Abed, Scott Fluhrer, Christian Forler, Eik List, Stefan Lucks, David McGrew, and Jakob Wenzel. Pipelineable on-line encryption. Cryptology ePrint Archive, Paper 2014/297, 2014. https://eprint.iacr.org/2014/297.

[2] Ange Albertini, Thai Duong, Shay Gueron, Stefan Kölbl, Atul Luykx, and Sophie Schmieg. How to abuse and fix authenticated encryption without key commitment. Cryptology ePrint Archive, Paper 2020/1456, 2020. https://eprint.iacr.org/2020/1456.

[3] Mila Anastasova, Panos Kampanakis, and Jake Massimo. Pq-hpke: Post-quantum hybrid public key encryption. Cryptology ePrint Archive, Paper 2022/414, 2022. https://eprint.iacr.org/2022/414.

[4] Richard Barnes, Karthikeyan Bhargavan, Benjamin Lipp, and Christopher A. Wood. Hybrid Public Key Encryption. RFC 9180, February 2022.

[5] Matt Campagna, Shay Gueron, Panos Kampanakis, Colm MacCarthaigh, Margaret Salter, and Daniel Simon. AWS' PUBLIC COMMENTS ON FIPS 197 - Advanced Encryption Standard (AES), 2021. https://csrc.nist.gov/CSRC/media/Projects/crypto-publication-review-project/documents/initial-comments/fips-197-initial-public-comments-2021.pdf.

[6] Matthew Campagna and Shay Gueron. Key management systems at the cloud scale. *Cryptography*, 3(3), 2019.

[7] Frank Denis, Fabio Enrico Renzo Scotoni, and Samuel Lucas. The AEGIS Family of Authenticated Encryption Algorithms. Internet-Draft draft-irtf-

cfrg-aegis-aead-03, Internet Engineering Task Force, April 2023. Work in Progress.

[8] Yevgeniy Dodis, Paul Grubbs, Thomas Ristenpart, and Joanne Woodage. Fast message franking: From invisible salamanders to encryptment. Cryptology ePrint Archive, Paper 2019/016, 2019. https://eprint.iacr.org/2019/016.

[9] Morris Dworkin. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. NIST Special Publication 800-38D, 2007. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf.

[10] Morris Dworkin. Recommendation for Block Cipher Modes of Operation: the XTS-AES Mode for Confidentiality on Storage Devices. NIST Special Publication 800-38E, 2010. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38e.pdf.

[11] Neils Ferguson. Authentication Weaknesses in GCM. Natl. Inst. Stand. Technol. [Web page], 2005. https://csrc.nist.gov/csrc/media/projects/block-cipher-techniques/documents/bcm/comments/cwc-gcm/ferguson2.pdf.

[12] Shay Gueron. Intel Advanced Encryption Standard (AES) New Instructions Set. Intel Whitepaper, 2010.

[13] Shay Gueron. Key committing AEADs. Cryptology ePrint Archive, Paper 2020/1153, 2020. https://eprint.iacr.org/2020/1153.

[14] Shay Gueron, Adam Langley, and Yehuda Lindell. AES-GCM-SIV: Nonce Misuse-Resistant Authenticated Encryption. RFC 8452, April 2019.

[15] Shay Gueron and Yehuda Lindell. Gcm-siv: Full nonce misuse-resistant authenticated encryption at under one cycle per byte. Cryptology ePrint Archive, Paper 2015/102, 2015. https://eprint.iacr.org/2015/102.

[16] Shay Gueron and Nicky Mouha. Simpira v2: A family of efficient permutations using the aes round function. Cryptology ePrint Archive, Paper 2016/122, 2016. https://eprint.iacr.org/2016/122.

[17] Julia Len, Paul Grubbs, and Thomas Ristenpart. Partitioning oracle attacks. Cryptology ePrint Archive, Paper 2020/1491, 2020. https://eprint.iacr.org/2020/1491.

[18] Sanketh Menda, Julia Len, Paul Grubbs, and Thomas Ristenpart. Context discovery and commitment attacks: How to break ccm, eax, siv, and more. Cryptology ePrint Archive, Paper 2023/526, 2023. https://eprint.iacr.org/2023/526.pdf.

[19] Mick Seaman. GCM Cipher Suites with Extended Packet Numbering, 2011. https://grouper.ieee.org/groups/802/1/files/public/docs2011/new-seaman-macsec-xpn-0711-v1.pdf.

[20] Bas Westerbaan and Christopher A. Wood. X25519Kyber768Draft00 hybrid post-quantum KEM for HPKE. Internet-Draft draft-westerbaan-cfrg-hpke-xyber768d00-02, Internet Engineering Task Force, May 2023. Work in Progress.